FILED - CLERK
U.S. DISTRICT COURT

UNITED STATES DISTRICT COURT
EASTERN DISTRICT OF TEXAS
BEAUMONT DIVISION

2004 JUN 29 PM 1:10

TX EASTERN-BEAUMONT

BY Barbara Thrail

FREESCALE SEMICONDUCTOR, INC.,

Plaintiff,

v.

ANALOG DEVICES, INC.,

Defendant.

CIVIL ACTION NO. 1:04CV0404

JURY TRIAL DEMANDED

COMPLAINT FOR DAMAGES AND INJUNCTIVE RELIEF

Plaintiff Freescale Semiconductor, Inc., by its attorneys, complains against Analog Devices, Inc., and alleges as follows:

Parties

1. Plaintiff Freescale Semiconductor, Inc. ("Freescale") is a corporation organized under the laws of the State of Delaware with its principal place of business at 6501 William Cannon Drive, Austin, Texas 78737. Freescale markets and sells semiconductor products throughout the United States, including within this District.

2. Defendant Analog Devices, Inc. ("ADI") is a corporation organized under the laws of the State of Massachusetts, with its principal place of business at One Technology Way, Norwood, MA 02062, and may be served by and through its agent for service of process, C.T. Corporation System, 350 N. St. Paul Street, Dallas, Texas 75201. ADI markets and sells semiconductor products throughout the United States, including within this District.

Jurisdiction and Venue

3. This is an action arising under the patent laws of the United States, 35 U.S.C. § 101 et seq. This Court has subject matter jurisdiction under 28 U.S.C. §§ 1331 and 1338(a).

4. Venue is proper in this judicial district under 28 U.S.C. §§ 1391(b), (c) and (d) and 1400(b).

The Patents

5. United States Patent No. 5,040,144, invented by Perry Pelley, entitled "Integrated Circuit with Improved Power Supply Distribution" (the "Pelley Patent"), was duly and legally issued by the United States Patent and Trademark Office on August 13, 1991. A copy of the Pelley Patent is attached hereto as Exhibit A.

6. United States Patent No. 4,758,945, invented by James J. Remedi, entitled "Method for Reducing Power Consumed by a Static Microprocessor" (the "Remedi Patent"), was duly and legally issued by the United States Patent and Trademark Office on July 19, 1988. A copy of the Remedi Patent is attached hereto as Exhibit B.

7. United States Patent No. 5,084,814, invented by John J. Vaglica, Jay A. Hartvigsen, and Rand L. Gray, entitled "Data Processor with Development Support Features" (the "Vaglica Patent"), was duly and legally issued by the United States Patent and Trademark Office on January 28, 1992. A copy of the Vaglica Patent is attached hereto as Exhibit C.

8. Freescale is the owner of all rights, title and interest in and to the Pelley, Remedi, and Vaglica Patents (the "Freescale Patents").

Background

9. The Freescale Patents cover inventions relating to electrical circuits, semiconductor processing and semiconductor chip design.

10. The Defendant has, without authorization, made, used, offered for sale, sold, and/or imported within the United States products covered by the Freescale Patents.

11. The Defendant has had actual and/or constructive notice and knowledge of the Freescale Patents. The filing of this Complaint also constitutes notice in accordance with 35 U.S.C. § 287. Despite such notice, the Defendant continues to make, use, offer for sale, sell, and/or import within the United States products covered by the Freescale Patents.

Count I

12. Plaintiff repeats and realleges the allegations in paragraphs 1-11.

13. On information and belief, the Defendant has infringed, and/or induced infringement of and/or contributed to the infringement of the Pelley Patent by making, using, offering for sale, selling, and/or importing within the United States products that incorporate the invention of the Pelley Patent.

14. As used in paragraph 13 above, the phrase "products that incorporate the invention of the Pelley Patent" does not include ADSP-21062 processors.

15. On information and belief, the Defendant's infringement of the Pelley Patent has been willful. The Defendant's continued infringement of the Pelley Patent has damaged and will continue to damage Plaintiff.

16. On information and belief, the Defendant's infringement of the Pelley Patent has caused and will continue to cause Plaintiff irreparable harm unless enjoined by the Court. Plaintiff has no adequate remedy at law.

Count II

17. Plaintiff repeats and realleges the allegations in paragraphs 1-11.

18. On information and belief, the Defendant has infringed, and/or induced infringement of and/or contributed to the infringement of the Remedi Patent by making, using, offering for sale, selling, and/or importing within the United States products that incorporate the invention of the Remedi Patent.

19. As used in paragraph 18 above, the phrase "products that incorporate the invention of the Remedi Patent" does not include ADSP-21XX family processors or products that incorporate an ADSP-21XX Processor or ADSP-21XX Processor Core that implement the power down or an equivalent feature.

20. On information and belief, the Defendant's infringement of the Remedi Patent has been willful. The Defendant's continued infringement of the Remedi Patent has damaged and will continue to damage Plaintiffs.

21. On information and belief, the Defendant's infringement of the Remedi Patent has caused and will continue to cause Plaintiff irreparable harm unless enjoined by the Court. Plaintiff has no adequate remedy at law.

Count III

22. Plaintiff repeats and realleges the allegations in paragraphs 1-11.

23. On information and belief, the Defendant has infringed, and/or induced infringement of and/or contributed to the infringement of the Vaglica Patent by making, using, offering for sale, selling, and/or importing within the United States products that incorporate the invention of the Vaglica Patent.

24. As used in paragraph 23 above, the phrase "products that incorporate the invention of the Vaglica Patent," does not include AD 6522, AD 6526, AD 6528, or products that include ARM7TDMI processor cores.

25. On information and belief, the Defendant's infringement of the Vaglica Patent has been willful. The Defendant's continued infringement of the Vaglica Patent has damaged and will continue to damage Plaintiff.

26. On information and belief, the Defendant's infringement of the Vaglica Patent has caused and will continue to cause Plaintiff irreparable harm unless enjoined by the Court. Plaintiff has no adequate remedy at law.

Prayer for Relief

WHEREFORE, Plaintiff respectfully requests that this Court enter judgment in its favor and grant the following relief:

A. Adjudge that the Defendant is infringing the Freescale Patents;

B. Adjudge that the Defendant's infringement of the Freescale Patents was willful, and that Defendant's continued infringement of the Freescale Patents is willful;

C. Enter an order preliminarily and permanently enjoining the Defendant from any further acts of infringement of the Freescale Patents;

D. Award Plaintiff damages in an amount adequate to compensate Plaintiff for the Defendant's infringement of the Freescale Patents, but in no event less than a reasonable royalty under 35 U.S.C. § 284;

E. Enter an order trebling any and all damages awarded to Plaintiff by reason of the Defendant's willful infringement of the Freescale Patents, pursuant to 35 U.S.C. § 284;

F. Enter an order awarding Plaintiff interest on the damages awarded and its costs pursuant to 35 U.S.C. § 284;

G. Enter an order finding that this is an exceptional case and award Plaintiff its reasonable attorneys' fees pursuant to 35 U.S.C. § 285; and

H. Award such other relief as the Court may deem appropriate and just under the circumstances.

Dated: June 29, 2004

J. Thad Heartfield

J. Thad Heartfield (Texas State Bar No. 09346800)
HEARTFIELD & MCGINNIS, L.L.P.
2195 Dowlen Road
Beaumont, TX 77706
Telephone: (409) 866-3318
Fascimile: (409) 866-5789

OF COUNSEL:

Victor G. Savikas
JONES DAY
555 West Fifth Street, Suite 4600
Los Angeles, CA 90013-1025
Telephone: (213) 243-2451
Facsimile: (213) 243-2539

David L. Witcoff
JONES DAY
77 West Wacker
Chicago, IL 60601-1692
Telephone: (312) 269-4259
Facsimile: (312) 782-8585

Michael J. Newton
JONES DAY
2727 N. Harwood Street
Dallas, Texas 75201-1515
Telephone: (214) 969-4869
Facsimile: (214) 969-5100

Attorneys for Plaintiff
FREESCALE SEMICONDUCTOR, INC.

United States Patent [19]
Pelley et al.

[11] Patent Number: 5,040,144
[45] Date of Patent: Aug. 13, 1991

[54] INTEGRATED CIRCUIT WITH IMPROVED POWER SUPPLY DISTRIBUTION

[75] Inventors: Perry Pelley, Austin, Tex.; Tim P. Egging, Colorado Springs, Colo.

[73] Assignee: Motorola, Inc., Schaumburg, Ill.

[21] Appl. No.: 442,268

[22] Filed: Nov. 28, 1989

[51] Int. Cl.[5] G11C 5/14
[52] U.S. Cl. 365/51; 365/63; 365/226; 365/230.03
[58] Field of Search 365/51, 63, 189.08, 365/230.03, 230.06, 226; 357/45

[56] References Cited

U.S. PATENT DOCUMENTS

4,458,297	7/1984	Stopper et al.	357/45
4,780,846	10/1988	Tanabe et al.	365/63
4,849,943	7/1989	Pennings	365/230.06

OTHER PUBLICATIONS

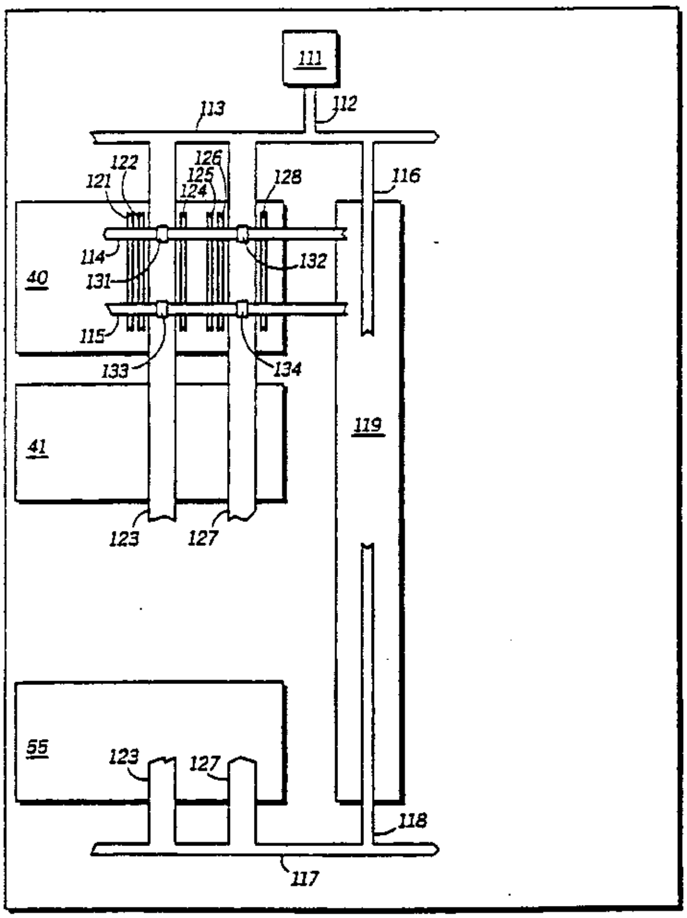
"Power Grid Image for Embedded Arrays", IBM Technical Disclosure Bulletin, vol. 32, No. 813 Jan. 1990 pp. 451-452.

Primary Examiner—Stuart N. Hecker
Assistant Examiner—Jack A. Lane
Attorney, Agent, or Firm—Paul J. Polansky

[57] ABSTRACT

An integrated circuit with reduced size through improved power supply distribution. A bonding pad supplies V_{SS} to an integrated circuit memory, which is distributed through a plurality of power supply lines in a first metal layer and a plurality of grid lines in a second metal layer intersecting at right angles. The plurality of grid lines are placed in unused spaced in the second metal layer and are coupled to the power supply lines in the first metal layer. Together the grid lines and the power supply lines provide an improved power supply by lowering the impedance from a point on the integrated circuit to V_{SS} supplied on the bonding pad. While this technique is ideally suited to memory devices because of the repetitive nature of blocks of memory cells, other types of integrated circuits can also utilize such a power supply distribution technique.

8 Claims, 3 Drawing Sheets



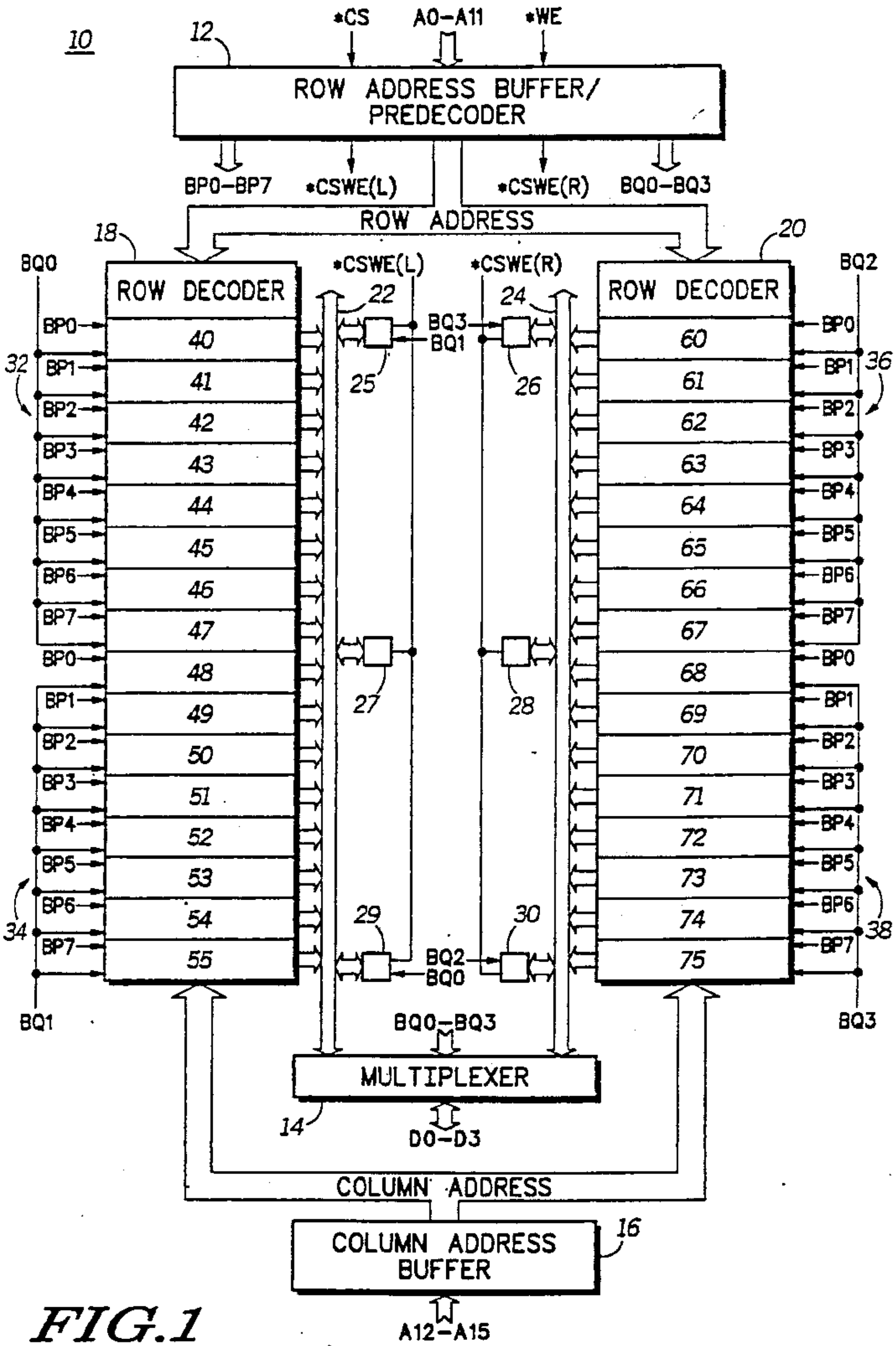
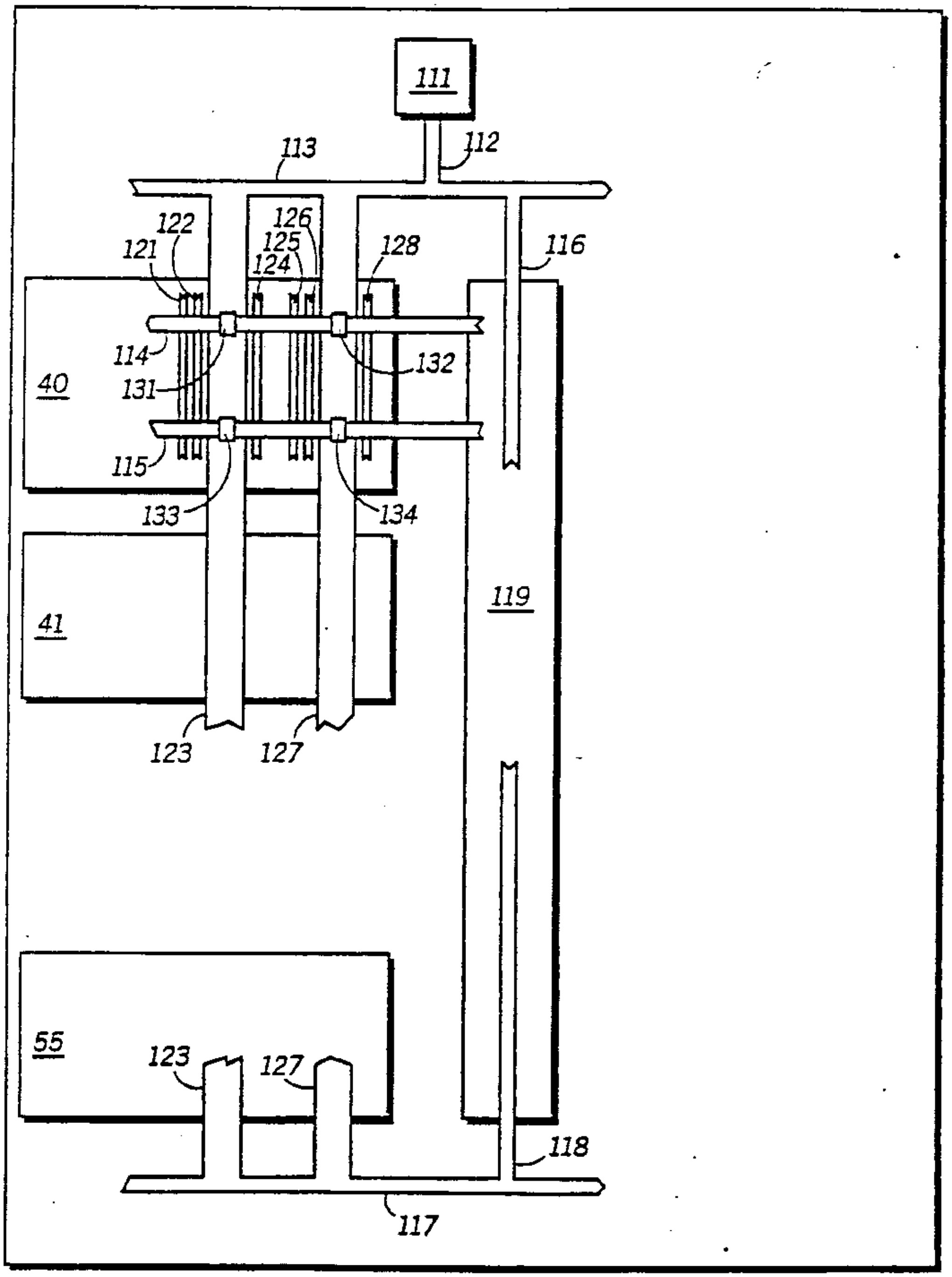

10
12
*CS
A0–A11
*WE
ROW ADDRESS BUFFER/ PREDECODER
BP0–BP7
*CSWE(L)
*CSWE(R)
BQ0–BQ3
ROW ADDRESS
18
20
BQ0
BQ2
ROW DECODER
ROW DECODER
*CSWE(L)
*CSWE(R)
22
24
BQ3
BQ1
25
26
27
28
29
30
BQ2
BQ0
32
34
36
38
BP0
BP1
BP2
BP3
BP4
BP5
BP6
BP7
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
BQ1
BQ3
BQ0–BQ3
MULTIPLEXER
14
D0–D3
COLUMN ADDRESS
COLUMN ADDRESS BUFFER
16
A12–A15

FIG.1



111
112
113
122
121
126
125
124
128
116
114
40
131
132
115
133
134
119
41
123
127
55
123
127
118
117

FIG.2

INTEGRATED CIRCUIT WITH IMPROVED POWER SUPPLY DISTRIBUTION

FIELD OF THE INVENTION

This invention relates generally to an integrated circuit with improved power supply distribution, and more particularly, to an integrated circuit memory with reduced size through improved power supply distribution.

BACKGROUND OF THE INVENTION

Metal oxide semiconductor (MOS) integrated circuits typically operate from a +5 volt power supply, V_{DD}, with a second power supply forming a ground, or V_{SS}, supply at 0 volts. In integrated circuit memories, these power supplies are each typically provided on one device pin. Each device pin is wire-bonded to a bonding pad on the integrated circuit itself. Although bond wires and package pins insert a series resistance and inductance between the bonding pad and the external power supply, it is useful to think of the power supply provided on the bonding pad as an ideal voltage source. One or more metal power supply lines are connected to each power supply bonding pad and route the power supply voltages throughout the integrated circuit. As the distance between functional circuitry and the bonding pad increases, the characteristics of the power supply provided to the functional circuitry become less ideal. Connecting to the ideal voltage source via a relatively thin metal line adds series resistance and capacitance between the connection point and the ideal voltage source. For the V_{SS} supply, when the current, I_{DD}, is at a peak, the voltage level on the V_{SS} lines the most distant from the bonding pad can rise and cause the functional circuitry to malfunction.

In integrated circuits with only one device pin for a given power supply, a main metal line, typically 0.01 inches wide for a large integrated circuit memory, is used to bus the power supply to all the functional circuitry. Smaller metal lines connect to the main metal line, but the main metal line is the primary conduit for routing a power supply from one side of the integrated circuit to another. The single-pin requirement fixes a maximum length between the bonding pad and the functional circuitry on the opposite end of the integrated circuit for a given size of the main metal line. Conversely, for a given length from the farthest functional circuitry to the bonding pad, the main metal line must be at least a given width to provide a power supply adequate to ensure proper operation.

Resistance through a conductor can be determined by the following formula:

$$R=(\rho^{*}L)/W$$

where ρ is the resistivity of the conductor, W is the width of the conductor, and L is the length of the conductor. Because of this basic relation, and since ρ and L are fixed, the width W of the metal conduit determines whether the resistance R is low enough to prevent incorrect operation during times of peak current. A technique used in the prior art has been to increase W until R becomes low enough to ensure correct operation. A problem with this approach, however, is that W must be so large, 0.01 inches typically, that the size of the integrated circuit increases substantially due to the width of the main power bus lines. The tradeoff between W and

integrated circuit size also results in a power supply that is barely good enough to avoid malfunction.

BRIEF DESCRIPTION OF THE INVENTION

Accordingly, it is an object of the present invention to provide an integrated circuit device with decreased size through improved power supply distribution.

It is another object of the present invention to provide an integrated circuit memory with with improved power supply distribution.

In carrying out these and other objects of the invention, there is provided, in one form, an integrated circuit with functional circuitry therein. The integrated circuit comprises a bonding pad, a plurality of horizontal lines, and a plurality of vertical lines. The bonding pad is for being coupled to a power supply voltage terminal. The plurality of horizontal lines are coupled to coupled to the bonding pad and to the functional circuitry. The plurality of vertical lines are coupled to the plurality of horizontal lines and intersect the horizontal lines at right angles. The plurality of horizontal lines and the plurality of vertical lines together couple the power supply voltage terminal to the functional circuitry.

These and other objects, features and advantages will be more clearly understood from the following detailed description taken in conjunction with the accompanying drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is a block diagram of an integrated circuit memory;

FIG. **2** is a plan view of portions of the integrated circuit memory of FIG. **1** illustrating a preferred embodiment of the present invention; and

FIG. **3** is a schematic layout of the preferred embodiment of FIG. **2** which forms a repeating structure over a plurality of pairs of memory cells.

DETAILED DESCRIPTION OF THE INVENTION

Shown in the FIG. **1** is a memory **10** generally comprising a row address buffer/predecoder **12**, a multiplexer **14**, a column address buffer **16**, a first row decoder **18**, a second row decoder **20**, a first set of global data lines **22**, a second set of global data lines **24**, global data line load sets **25,26,27, 28,29**, and **30**, a first array **32**, a second array **34**, a third array **36**, and a fourth array **38**. First set of global data lines **22** comprises four global data line pairs, each pair including a true global data line and a complementary global data line. Second set of global data lines **24** comprises four global data line pairs, each pair including a true global data line and a complementary global data line. First array **32** comprises memory blocks **40,41,42,43,44,45,46**, and **47**. Second array **34** comprises memory blocks **48,49,50,51,52,53,54**, and **55**. Third array **36** comprises memory blocks **60,61,62,63,64,65,66**, and **67**. Fourth array **38** comprises memory blocks **68,69,70,71,72,73,74**, and **75**.

Row address buffer/predecoder **12** receives row address signals A0, A1, A2,A3,A4,A5,A6,A7,A8,A9,A10, and A11, chip select signal *CS, and write mode signal *WE. Row address buffer/predecoder **12** provides memory block select signals BP0,BP1,BP2,BP3,BP4,BP5,BP6, and BP7, chip select/write signals *CSWE(L) and *CSWE(R), array select signals BQ0,BQ1,BQ2, and BQ3, and a buffered

row address signal. An asterisk (*) in front of a signal designation indicates that signal is active at a logic low. Column address buffer 16 receives column address signals A12,A13,A14, and A15, and outputs a buffered column address. Multiplexer 14 receives signals BQ0–BQ3 and is coupled to sets of global data lines 22 and 24. In a read mode of memory 10, multiplexer 14 receives global data line signals GDL(0) and *GDL(0), GDL(1) and *GDL(1),GDL(2) and *GDL(2), and GDL(3) and *GDL(3) from first set of global data lines 22, and global data line signals GDL(4) and *GDL(4), GDL(5) and *GDL(5),GDL(6) and *GDL(6), and GDL(7) and *GDL(7) from second set of global data lines 24. Multiplexer 14 outputs data signals D0,D1,D2, and D3 in the read mode, and receives data input signals D0–D3 in a write mode of memory 10. First row decoder 18 receives the buffered row address and selectively provides 64 global word line driver signals and a buffered row address signal to memory blocks 40–55. Second row decoder 20 receives the buffered row address and selectively provides 64 global word line driver signals and a buffered row address signal to memory blocks 60–75.

Array 32 is located immediately above array 34, and to the left of arrays 36 and 38. Memory blocks within array 32 are placed sequentially below one another, with memory block 40 at the top of array 32, memory block 41 below memory block 40, etc. Memory block 47 is at the bottom of array 32. Memory blocks within array 34 are placed sequentially below one another, with memory block 48 at the top of array 34, memory block 49 below memory block 48, etc. Memory block 55 is at the bottom of array 34. Array 36 is located immediately above array 38. Memory blocks within array 36 are placed sequentially below one another, with memory block 60 at the top of array 36, memory block 61 below memory block 60, etc. Memory block 67 is at the bottom of array 36. Memory blocks within array 38 are placed sequentially below one another, with memory block 68 at the top of array 38, memory block 69 below memory block 68, etc. Memory block 75 is at the bottom of array 38. First set of global data lines 22 begins at the top of array 32 and runs below the bottom of array 34, where it connects with multiplexer 14. Second set of global data lines 24 begins at the top of array 36 and runs below the bottom of array 38, where it connects with multiplexer 14.

Global data line load set 25 connects to first set of global data lines 22 at the top of array 32. Below global data line load set 25 memory blocks 40–47 in first array 32 connect to first set of global data lines 22. Below first array 32 global data line load set 27 connects to first set of global data lines 22. Below global data line load set 27 memory blocks 48–55 in second array 34 connect to first set of global data lines 22. Below second array 34 global data line load set 29 connects to first set of global data lines 22. Multiplexer 14 connects to the first set of global data lines 22. In the read mode, multiplexer 14 receives signals GDL(0)/*GDL(0)–GDL(3)/*GDL(3) on the four global data line pairs contained in first set of global data lines 22.

Global data line load set 26 connects to the second set of global data lines 24 at the top of array 36. Below global data line load set 26 memory blocks 60–67 in third array 36 connect to second set of global data lines 24. Below third array 36 global data line load set 28 connects to second set of global data lines 24. Below global data line load set 28 memory blocks 68–75 in

fourth array 38 connect to the second set of global data lines 24. Below fourth array 38 global data line load set 30 connects to the second set of global data lines 24. Multiplexer 14 connects to second set of global data lines 24. In the read mode, multiplexer 14 receives signals GDL(4)/*GDL(4)–GDL(7)/*GDL(7) on the four global data line pairs contained in second set of global data lines 24.

Global data line load set 25 receives array select signal BQ1 and chip select/write signal *CSWE(L). Global data line load set 27 receives chip select/write signal *CSWE(L). Global data line load set 29 receives array select signal BQ0 and chip select/write signal *CSWE(L). Global data line load set 26 receives array select signal BQ3 and chip select/write signal *CSWE(R). Global data line load set 28 receives chip select/write signal *CSWE(R). Global data line load set 30 receives array select signal BQ2 and chip select/-write signal *CSWE(R).

Array 32 is selected when signal BQ0 is a logic high. One of memory blocks 40–47 is selected when array 32 is selected. The particular memory block 40–47 within array 32 which is selected when its corresponding signal of signals BP0–BP7 is a logic high. In array 32, memory blocks 40–47 connect to first row decoder 18 and to column address buffer 16. Memory blocks 40–47 receive memory block select signals BP0–BP7, respectively. Each of memory blocks 40–47 receives array select signal BQ0 from row address buffer/decoder 12, and has four data output pairs coupled to first set of global data lines 22. In the read mode, a selected memory block provides output signals onto first set of global data lines 22 via the four data output pairs coupled between the selected memory block and first set of global data lines 22. Similarly, in the write mode, multiplexer 14 outputs signals onto the first set of global data lines 22 and the selected memory block reads these signals.

Array 34 is selected when signal BQ1 is a logic high. One of memory blocks 48–55 is selected when array 34 is selected. The particular memory block 48–55 within array 34 which is selected is selected when its corresponding signal of signals BP0–BP7 is a logic high. In array 34, memory blocks 48–55 connect to first row decoder 18 and to column address buffer 16. Memory blocks 48–55 receive memory block select signals BP0–BP7, respectively. Each of memory blocks 48–55 receives array select signal BQ1 from row address buffer/decoder 12, and has four data output pairs coupled to first set of global data lines 22. In the read mode, a selected memory block provides output signals onto first set of global data lines 22 via the four data output pairs coupled between the selected memory block and first set of global data lines 22. Similarly, in the write mode, multiplexer 14 outputs signals onto the first set of global data lines 22 and the selected memory block reads these signals.

Array 36 is selected when signal BQ2 is a logic high. One of memory blocks 60–67 is selected when array 36 is selected. The particular memory block 60–67 within array 36 which is selected is selected when its corresponding signal of signals BP0–BP7 is a logic high. In array 36, memory blocks 60–67 connect to second row decoder 20 and to column address buffer 16. Memory blocks 60–67 receive memory block select signals BP0–BP7, respectively. Each of memory blocks 60–67 receives array select signal BQ2 from row address buffer/decoder 12, and has four data output pairs coupled

to second set of global data lines 24. In the read mode, a selected memory block provides output signals onto second set of global data lines 24 via the four data output pairs coupled between the selected memory block and second set of global data lines 24. Similarly, in the write mode, multiplexer 14 outputs signals onto the second set of global data lines 24 and the selected memory block reads these signals.

Array 38 is selected when signal BQ3 is a logic high. One of memory blocks 68-75 is selected when array 38 is selected. The particular memory block 68-75 within array 38 which is selected is selected when its corresponding signal of signals BP0-BP7 is a logic high. In array 38, memory blocks 68-75 connect to second row decoder 20 and to column address buffer 16. Memory blocks 68-75 receive memory block select signals BP0-BP7, respectively. Each of memory blocks 68-75 receives array select signal BQ3 from row address buffer/decoder 12, and has four data output pairs coupled to second set of global data lines 24. In the read mode, a selected memory block provides output signals onto second set of global data lines 24 via the four data output pairs coupled between the selected memory block and second set of global data lines 24. Similarly, in the write mode, multiplexer 14 outputs signals onto the second set of global data lines 24 and the selected memory block reads these signals.

In operation, memory 10 allows memory cells located within memory blocks 40-55 and 60-75 to be read from and written to. During write cycles, multiplexer 14 receives data signals D0-D3 and supplies them to the appropriate memory block based on the address A0-A15. In a write mode, when a memory block on a left side of memory 10 comprising blocks in arrays 32 or 34 is being written to, *CSWE(L) is low and global data line load sets 25, 27, and 29 are disabled. Global data line load sets 26, 28, and 30 are enabled, however, to prevent the second set of global data lines from assuming an indeterminate state. When a memory block on a right side of memory 10 is being written to, *CSWE(R) is low and global data line load sets 26, 28, and 30 are disabled. Global data line load sets 25, 27, and 29 are enabled, however, to prevent the first set of global data lines from assuming an indeterminate state.

In the read mode, memory 10 provides four bits of data represented by data signals D0-D3 selected by address signals A0-A15. Column address buffer 16 buffers incoming address signals A12-A15 and outputs them to memory blocks 40-55 and 60-75. Row address buffer/predecoder 12 decodes row address lines A0-A11, chip select signal *CS, and write mode signal *WE. In response, it supplies signals BP0-BP7, BQ0-BQ3, *CSWE(L) and *CSWE(R), and a row address to row decoders 18 and 20. Signals BP0-BP7 select one of eight memory blocks of each array 32, 34, 36, and 38. Signals BQ0-BQ3 select which one of the four arrays 32, 34, 36, and 38 is selected. Together, signals BP0-BP7 and signals BQ0-BQ3 select one memory block of 32 memory blocks of the set 40-55 and 60-75. *CSWE(L) is true if both *CS and *WE are true and a left side comprising arrays 32 and 34 is selected, and indicates that memory 10 is in the write mode, that memory 10 is active, and that the global data line loads connected to first set of global data lines 22 should be disabled. *CSWE(R) is true if both *CS and *WE are true and a right side comprising arrays 36 and 38 is selected, and indicates that memory 10 is in the write mode, that memory 10 is active, and that the global data line loads connected to second set of global data lines 24 should be disabled.

The buffered row address is input to first row decoder 18 and second row decoder 20. In response to receiving the buffered row address, first row decoder 18 drives 64 global word lines to memory blocks 40-55 and second row decoder 20 drives 64 global word lines to memory blocks 60-75. The word lines, along with the column address and the buffered row address signal, are further decoded in the memory blocks themselves. After a particular memory block is selected by BP0-BP7 and BQ0-BQ3, the memory block combines the 64 global word lines and the buffered row address signal and drives 128 local word lines. The memory block decodes the column address along with the selected word line and selects one memory cell for each of four pairs of global data lines. The four memory cells output four data bits and the complements of the four data bits onto the first set of global data lines 22 if the memory block selected is located in either first array 32 or second array 34, or outputs four data bits and the complements of the four data bits onto the second set of global data lines 24 if the memory block selected is located in first array 36 or second array 38. Multiplexer 14 receives signals GDL(0)/*GDL(0)-GDL(3)/*GDL(3) from first set of global data lines 22 and signals GDL(4)/*GDL(4)-GDL(7)/*GDL(7) from second set of global data lines 24 and forms and buffers outputs DO-D3 in response. Multiplexer 14 receives signals BQ0-BQ3 and in response derives D0- D3 from signals GDL(0)/*GDL(0)-GDL(3)/*GDL(3) if a memory block in the first array 32 or second array 34 is selected by BQ0 or BQ1, respectively, or from signals GDL(4)/*GDL(4)-GDL(7)/*GDL(7) if a memory block in the third array 36 or fourth array 38 is selected by BQ2 or BQ3, respectively.

When a read access takes place, a selected memory block outputs signals on four global data line pairs. Each global data line pair appears as a pair of transmission lines. The selected memory block outputs signals on each long transmission line using a differential transconductance amplifier, which receives a sensed differential voltage from a selected memory cell and outputs a differential current in response. The global data line loads source the current and thereby convert the output of the transconductance amplifiers to a voltage, so that the signals GDL(0)/*GDL(0)-GDL(7)/*GDL(7) form eight differential voltage pairs.

The particular global data line load sets selected depends on which array contains a memory block which is selected. If first array 32 or second array 34 is selected during a read access, signal *CSWE(L) is high. During a memory access in which *CSWE(L) is high, global data line load set 27 is always selected. Which one of global data line load set 25 and global data line load set 29 is selected is determined by whether a memory block in first array 32 or second array 34 is selected. If a memory block in first array 32 is selected, BQ0 is high, BQ1-BQ3 are low, global data line load set 25 is enabled, and global data line load set 29 is disabled. If a memory block in second array 34 is selected, BQ1 is high, BQ0 and BQ2-BQ3 are low, global data line load set 29 is enabled, and global data line load set 25 is disabled.

Similarly, if third array 36 or fourth array 38 is selected during a read access, signal *CSWE(R) is high. During a memory access in which *CSWE(R) is high,

global data line load set 28 is always selected. Which of global data line load set 26 and global data line load set 30 is selected is determined by whether a memory block in third array 36 or fourth array 38 is selected. If a memory block in third array 36 is selected, BQ2 is high, BQ0, BQ1, and BQ3 are low, global data line load set 26 is enabled, and global data line load set 30 is disabled. If a memory block in fourth array 38 is selected, BQ3 is high, BQ0-BQ2 are low, global data line load set 30 is enabled, and global data line load set 26 is disabled.

FIG. 2 shows a plan view of a memory 10' which represents portions of memory 10 of FIG. 1. Memory 10' comprises memory blocks 40, 41, and 55 of FIG. 1, a bonding pad 111, a metal interconnect line 112, a metal power supply bus 113, a metal power supply line 114, a metal power supply line 115, a metal power supply line 116, a metal power supply bus 117, a metal power supply line 118, and a circuit 119. Shown in FIG. 2 are representative portions of memory block 40 useful in understanding the present invention, including a metal local word line 121, a metal global word line 122, a metal grid line 123, a metal local word line 124, a metal local word line 125, a metal global word line 126, a metal grid line 127, a metal local word line 128, a contact 131, a contact 132, a contact 133, and a contact 134. Grid lines 123 and 127 extend through all memory blocks 41-55 on a left half of memory 10', and memory blocks 41 and 55 are included to illustrate the routing. Other memory blocks 42-54, and 60-75 on a right side of memory 10 of FIG. 1 have similar routings of metal lines, but are omitted for ease of illustration.

In memory 10', bonding pad 111 provides an interconnection point for an external negative power supply voltage terminal V_{SS}. As earlier indicated, V_{SS} is a ground reference power supply and is approximately 0 volts. Bonding pad 111 couples to the external power supply voltage terminal through a bond wire connecting bonding pad 111 to a point on a lead frame, the lead frame providing an external pin or connection point to couple to the external supply. It is useful to consider V_{SS} supplied on bonding pad 111 as being an ideal voltage source. Power supply bus 113 is coupled to bonding pad 111 through interconnect line 112. Power supply lines 114, 115, and a plurality of power supply lines not shown are coupled to bonding pad 111 via power supply bus 113, and provide V_{SS} to memory block 40 in a horizontal direction. Grid lines 123 and 127 and a plurality of other grid lines not shown, running in a vertical direction, connect to power supply lines like power supply lines 114 and 115. Grid line 123 connects to power supply line 114 through contact 131, and to power supply line 115 through contact 133. Grid line 127 connects to power supply line 114 through contact 132, and to power supply line 115 through contact 134. Collectively, power supply lines 114, 115, and the plurality of horizontal metal lines not shown, along with grid lines 123, 127, and the plurality of grid lines not shown, effectively form a grid for providing V_{SS} to memory block 40. The grid approximates a metal plate, which substantially decreases resistance between a given point and the bonding pad below the resistance provided by a long metal line.

Grid lines 123 and 127 extend downward through memory 10' through all memory blocks on the left half of memory 10'. At the bottom of memory 10', grid lines 123 and 127 connect to power supply bus 117, which provides a power bus for the bottom of memory 10'. Power supply line 116 connects to power supply bus 113 and supplies power to a top half of circuit 119. Power supply line 118 connects to power supply bus 117 and supplies power to a bottom half of circuit 119. Additionally metal lines like power supply lines 114 and 115 also supply power to circuit 119.

In memory 10', a memory cell is located at each intersection of a local word line and a pair of bit lines. The global word lines are routed through each memory block in a given half of memory 10'. For example, global word lines 122 and 126 are routed to each of memory blocks 40-55. Inside the memory blocks, local word lines such as local word lines 121, 124, 125, and 128 are coupled to the global word lines via decoding logic.

Electrically, connecting the plurality of grid lines, of which grid lines 123 and 127 are examples, to the plurality of power supply lines, of which power supply lines 114 and 115 are examples, reduces resistance between a memory cell and the V_{SS} supply, all else equal. The distribution provided by the resulting grid closely approximates a plate, which provides the lowest resistance between two points within a bounded area. In the plate, W is maximized in relation to L, thereby minimizing R. The grid provided by lines 114, 115, 123, 127, and others not shown covers memory block 40. In other memory blocks a plurality of power supply lines and a plurality of grid lines form similar grids. In addition, a plurality of power supply lines in memory block 40 like power supply lines 114 and 115 shown in FIG. 2 connect the grid to other circuits like circuit 119. In this case, circuit 119 also benefits from a lowered resistance to V_{SS}. A grid, not shown in FIG. 2, is similarly formed by power supply lines and grid lines on the right half of the memory, and provides an improved power supply to memory blocks 60-75.

FIG. 3 shows in greater detail a layout cell 140 of some of the metal lines of memory block 40 of FIG. 2 over a pair of memory cells, which forms a repeating structure. Layout cell 140 comprises a metal local word line 121', a metal global word line 122', a metal grid line 123', a metal local word line 124', a metal bit line 141, a metal inverted bit line 142, a partial contact 143, a partial contact 144, a partial contact 145, and a partial contact 146. Lines 121', 122', 123', and 124' correspond to lines 121, 122, 123, and 124 of FIG. 2 and constitutes segments thereof, and hence are numbered similarly for convenience.

Underneath layout cell 140 are two memory cells, one on each half of a line running in the vertical direction down the middle of layout cell 140. One memory cell is rotated 180 degrees in reference to another memory cell to preserve a symmetry between the two. Metal lines through layout cell 140 form a regular pattern only every two memory cells in the horizontal direction. Partial contact 143 is part of a contact in an adjoining cell, which in layout cell 140 corresponds to partial contact 145. Partial contact 144 is part of a contact in the adjoining cell, which in layout cell 140 corresponds to partial contact 146. Unlike contacts 131-134 of FIG. 2, contacts formed by partial contacts 143-146 make an electrical connection between a metal layer and an underlying circuitry layer, instead of between two metal layers.

As with corresponding word line 122 in FIG. 2, global word line 122' routes a decoded signal selecting a word or row to each memory block on a given half of memory 10 of FIG. 1. In contrast, local word lines 121' and 124' are actually used to select individual memory

cells. It is important to note the symmetry present between the layout cells to understand how the global word lines and the local word lines are used. Every sixteen layout cells in the vertical direction, local word lines **121'** and **124'** couple to an underlying polysilicon layer. The polysilicon layer further routes the local word line signals into each of the sixteen memory cells. For example, both memory cells in layout cell **140** are coupled to local word lines **121'** and **124'** by polysilicon lines. In a preferred embodiment, the polysilicon lines for a given sixteen memory cells are connected to the next sixteen memory cells and to the previous sixteen memory cells to lower impedance.

A selected memory cell is located at an intersection of an active word line and an active bit line pair, as determined by row and column decoding, respectively. Bit lines **141** and **142** are used to sense the contents of memory cells which are selected on active rows by word lines. Bit lines **141** and **142** are located in a first metal layer. In contrast, lines **121'**, **122'**, **123'**, and **124'** are located in a second metal layer. In the vertical direction after every four pairs of memory cells, grid line **123** connects to a horizontal power supply line, such as power supply line **114** or **115** of FIG. 2, to form the grid. These power supply lines in turn are coupled to underlying memory cells to provide V_{SS}, so that the grid lines do not themselves provide V_{SS} to the memory cells. In a preferred embodiment the second metal layer is above the first metal layer, but the order of the layers could be reversed. Grid line **123'** is placed in unused space in the second metal layer to allow the grid to be formed without sacrificing space in either layer. As pointed out earlier, use of grid lines like grid line **123'** improves power supply distribution so that wide metal V_{SS} bus lines running from the top end of the memory to the bottom end are no longer needed. Instead, power supply bus **113** and power supply bus **117** can be made much narrower. Use of the grid allows a smaller size for the memory, because large, 0.01 inch V_{SS} bus lines are no longer needed to provide a given resistance to the bonding pad.

It should be apparent by now that an integrated circuit memory with reduced size through the use of an improved power supply distribution apparatus has been shown. Although the grid apparatus is well suited for memories because of their repetitive cell structure, it could be used for other types of integrated circuits to provide improved power supply distribution. It should also be apparent that the grid apparatus could also be used for another power supply, such as V_{DD}.

While the invention has been described in the context of a preferred embodiment, it will be apparent to those skilled in the art that the present invention may be modified in numerous ways and may assume many embodiments other than that specifically set out and described above. Accordingly, it is intended by the appended claims to cover all modifications of the invention which fall within the true spirit and scope of the invention.

We claim:

1. A memory having a plurality of subarrays aligned in a first direction, comprising:
   - a plurality of metal decoding lines crossing the plurality of subarrays in the first direction, said plurality of metal decoding lines formed in a first predetermined layer;
   - a plurality of metal power supply lines crossing said plurality of subarrays in a second direction, said second direction substantially orthogonal to said first direction, said plurality of metal power supply lines formed in a second predetermined layer different from said first predetermined layer; and
   - a plurality of metal grid lines between said metal decoding lines and intersecting and coupled to said metal power supply lines, said plurality of metal grid lines formed in said first predetermined layer.

2. The memory of claim **1** wherein said plurality of metal decoding lines comprises a plurality of global word lines and a plurality of local word lines.

3. The memory of claim **1** wherein said metal power supply lines and said metal grid lines provide a power supply voltage terminal to the memory.

4. An integrated circuit comprising a plurality of memory blocks, each memory block comprising a plurality of memory cells, comprising:
   - a bonding pad, for providing an interconnection point to a power supply voltage terminal, said power supply voltage terminal external to the integrated circuit;
   - a plurality of horizontal lines, coupled to said bonding pad and located within at least one corresponding memory block, and formed in a first layer of the integrated circuit memory, for coupling said power supply voltage terminal to the plurality of memory cells within said at least one corresponding memory block;
   - a first plurality of vertical lines formed in a second layer different from said first layer of the integrated circuit memory; and
   - a second plurality of vertical lines, crossing and coupled to the plurality of horizontal lines, and located within a corresponding memory block, and formed in said second layer of the integrated circuit memory, for reducing an impedance between the plurality of memory cells and said power supply voltage terminal.

5. The integrated circuit memory of claim **4**, wherein said plurality of vertical lines and said plurality of horizontal lines are metal.

6. The integrated circuit memory of claim **4**, wherein said power supply voltage terminal is a negative power supply voltage terminal.

7. The memory of claim **1** wherein each subarray further comprises a plurality of layout cells, each layout cell comprising a segment of a corresponding metal decoding line of said plurality of metal decoding lines and a segment of a corresponding grid line of said plurality of grid lines.

8. The memory of claim **7**, wherein a predetermined number of layout cells is arranged in the first direction between adjacent power supply lines of said plurality of power supply lines.

* * * * *

UNITED STATES PATENT AND TRADEMARK OFFICE

CERTIFICATE OF CORRECTION

PATENT NO. : 5,040,144

DATED : August 13, 1991

INVENTOR(S) : Perry Pelley et al.

It is certified that error appears in the above-identified patent and that said Letters Patent is hereby corrected as shown below:

Column 10, line 21, after "circuit", replace "comprising" with --memory with--.

Signed and Sealed this

Twenty-third Day of February, 1993

Attest:

STEPHEN G. KUNIN

Attesting Officer *Acting Commissioner of Patents and Trademarks*

United States Patent [19]
Remedi

[11] **Patent Number:** **4,758,945**
[45] **Date of Patent:** **Jul. 19, 1988**

[54] **METHOD FOR REDUCING POWER CONSUMED BY A STATIC MICROPROCESSOR**

[75] Inventor: **James J. Remedi**, Austin, Tex.

[73] Assignee: **Motorola, Inc.**, Schaumburg, Ill.

[21] Appl. No.: **65,292**

[22] Filed: **Aug. 9, 1979**

[51] **Int. Cl.⁴** **G06F 1/04**
[52] **U.S. Cl.** **364/200**
[58] **Field of Search** ... 364/200 MS File, 900 MS File; 365/227

[56] **References Cited**

U.S. PATENT DOCUMENTS

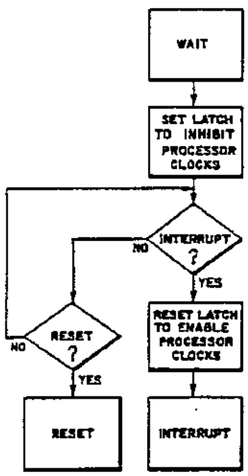
3,411,147	11/1968	Packard	364/200
3,535,560	10/1970	Cliff	364/200
3,736,569	5/1973	Bouricius et al.	364/200
3,855,577	12/1974	Vandierendonck	364/200
3,941,989	3/1976	McLaughlin et al.	235/156
4,030,079	6/1977	Bennett et al.	364/200
4,151,611	4/1979	Sugawara et al.	365/227
4,158,230	6/1979	Washizuka et al.	364/708
4,191,998	3/1980	Carmody	364/200

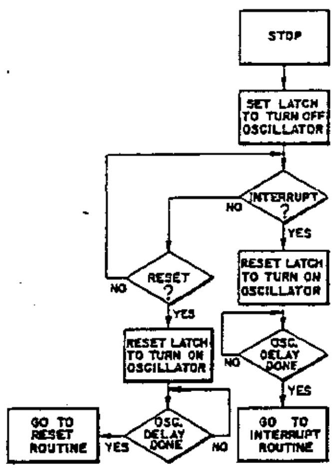
Primary Examiner—David Y. Eng
Attorney, Agent, or Firm—John A. Fisher; Jeffrey Van Myers

[57] **ABSTRACT**

In response to a software instruction, a static microprocessor is placed in a low current mode by disabling clock pulse generation. Means are provided for disabling a master oscillator when a STOP instruction is decoded. Additional means are provided for inhibiting clock pulses when a WAIT instruction is decoded without disabling the master oscillator. Clock pulse generation is again enabled upon receipt of a reset or interrupt signal.

6 Claims, 3 Drawing Sheets





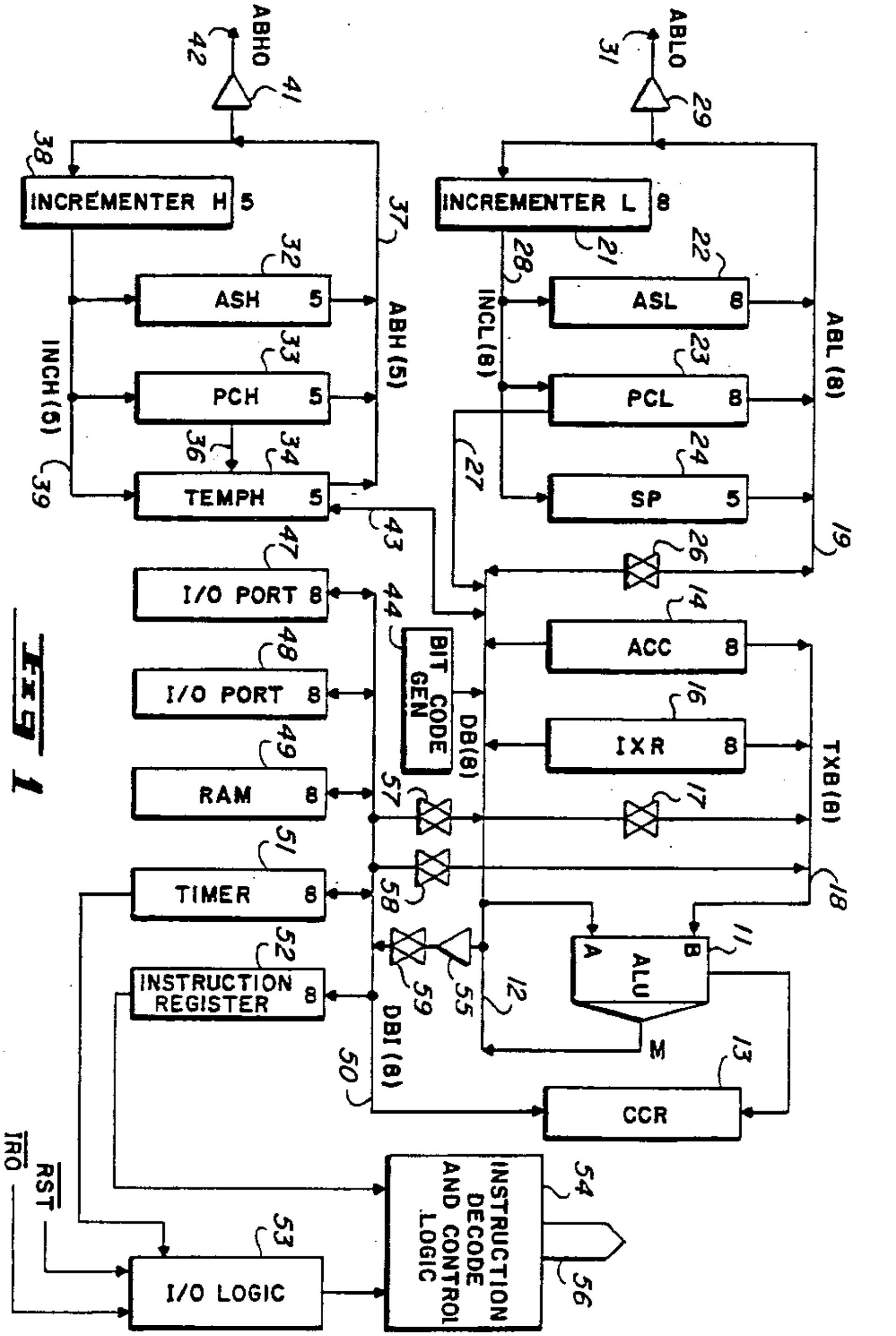

ABHO
ABLO
INCREMENTER H 5
INCREMENTER L 8
ASH 5
PCH 5
TEMPH 5
ASL 8
PCL 8
SP 5
INCH(5)
ABH(5)
INCL(8)
ABL(8)
I/O PORT 8
I/O PORT 8
RAM 8
TIMER 8
INSTRUCTION REGISTER 8
BIT CODE GEN
DB(8)
DBI(8)
ACC 8
IXR 8
TXB(8)
A
B
ALU
M
CCR
INSTRUCTION DECODE AND CONTROL LOGIC
I/O LOGIC
RST
IRQ

Fig. 1

Fig 2

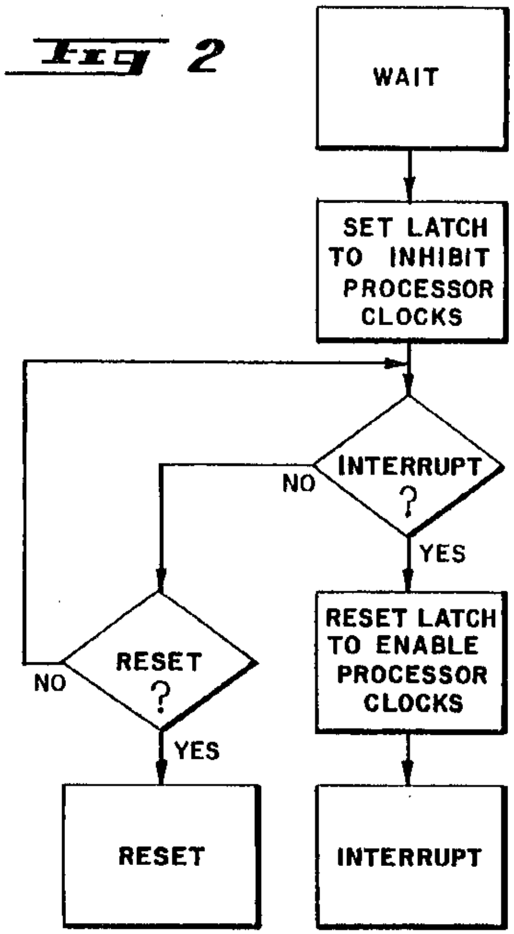
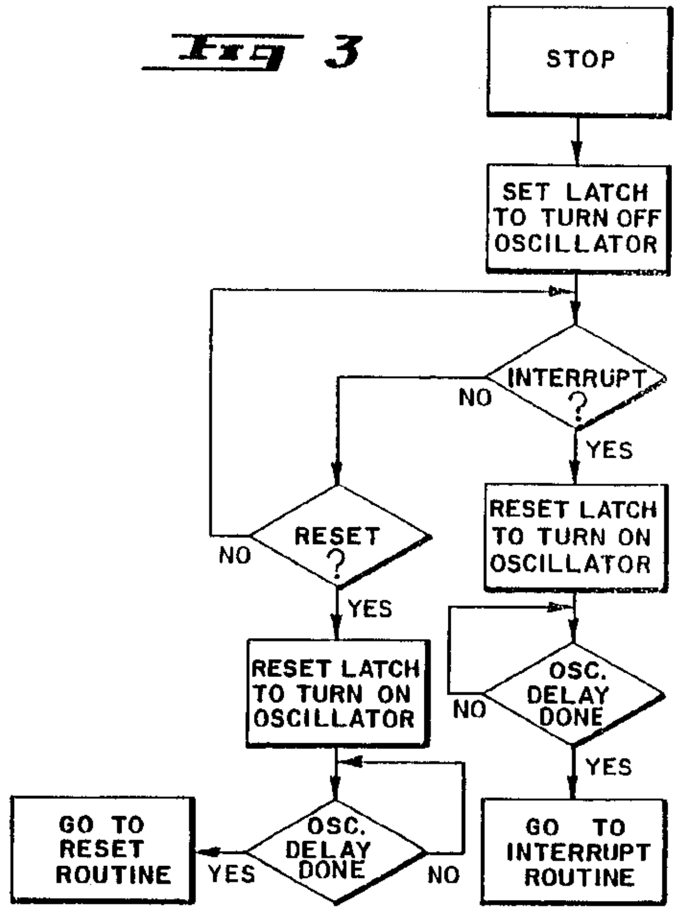
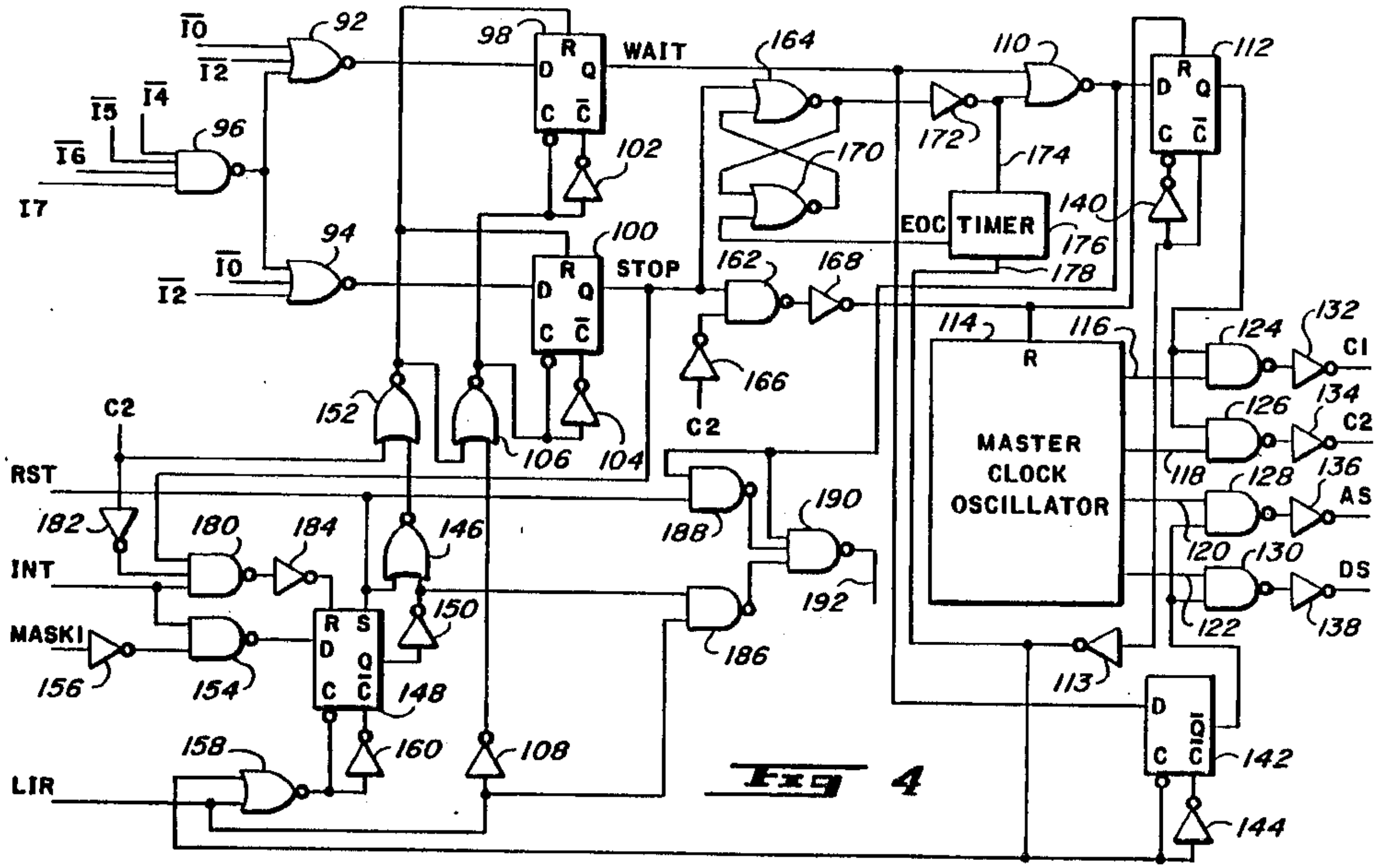

WAIT
SET LATCH TO INHIBIT PROCESSOR CLOCKS
INTERRUPT ?
NO
YES
RESET ?
NO
YES
RESET LATCH TO ENABLE PROCESSOR CLOCKS
RESET
INTERRUPT

Fig 3


STOP
SET LATCH TO TURN OFF OSCILLATOR
INTERRUPT ?
NO
YES
RESET ?
NO
YES
RESET LATCH TO TURN ON OSCILLATOR
RESET LATCH TO TURN ON OSCILLATOR
OSC. DELAY DONE
NO
YES
OSC. DELAY DONE
YES
NO
GO TO RESET ROUTINE
GO TO INTERRUPT ROUTINE



I0
I2
I4
I5
I6
I7
92
96
94
98
R
D
Q
C
WAIT
102
100
STOP
104
106
152
146
150
148
R S
D Q
C C
160
108
158
LIR
RST
INT
MASKI
C2
182
180
184
156
154
164
170
162
168
166
172
174
110
112
EOC
TIMER
176
178
140
114
116
MASTER
CLOCK
OSCILLATOR
118
120
122
124
126
128
130
132
134
136
138
CI
C2
AS
DS
188
190
192
186
113
142
144

Fig 4

METHOD FOR REDUCING POWER CONSUMED BY A STATIC MICROPROCESSOR

CROSS REFERENCE TO RELATED APPLICATIONS

(1) U.S. patent application Ser. No. 065,293 filed of even date herewith entitled "Apparatus for Reducing Power Consumed by a Static Microprocessor" and assigned to the assignee of the present invention.

(2) U.S. patent application Ser. No. 065,294 filed of even date herewith entitled "CMOS Microprocessor Architecture" and assigned to the assignee of the present invention.

BACKGROUND OF THE INVENTION

1. Field of the Invention

This invention relates generally to microcomputers and, more particularly, to a method for reducing the power consumed by static microprocessors.

2. Description of the Prior Art

Microcomputers are sophisticated, general purpose logic devices which can be programmed to perform a wide variety of useful control functions in industrial and communications equipment, large scale and medium scale computer peripheral and terminal hardware, automobiles and other transportation media, amusement and educational devices, household appliances and other consumer goods, and the like. Generally, an entire spectrum of microcomputers is presently available in the commercial marketplace. As the speed of operation increases, the more valuable and more versatile the microcomputer becomes since it is capable of controlling the given operation more efficiently and more accurately, of controlling a greater number of operations simultaneously, and of controlling operations requiring relatively fast response times.

The throughput of any given microcomputer is a function of, among other things, the number of machine cycles required to execute a given set of instructions. In the course of designing any computer system, and in particular a microcomputer, a set of instructions is selected which will provide the anticipated program requirements for the projected market in which the computer system is to be used. The microprocessor, or processor component of a single chip microcomputer, executes each instruction as a sequence of machine cycles, with the more complex instructions consuming a greater number of machine cycles.

The operation of the internal circuitry of the microprocessor is synchronized by means of a master clock signal applied to the microprocessor. The master clock signal may actually comprise two or even four clock components; i.e., the microprocessor clock may be two phase or four phase. During the basic clock cycle known as the machine cycle, a number of internal processor related operations may take place simultaneously including the transfer of digital information from a bus to a register or vice versa, between certain registers, from an address or data buffer to a bus or vice versa, and so forth. Additionally, the individual conductors of a bus may each be set to a predetermined logic level, or the contents of a register may be set to a predetermined logic level.

It is also desirable, particularly with respect to microcomputers intended for marketing in the middle to low end of the price scale, to minimize the computer chip size as much as possible.

Static microprocessors implemented with complementary MOS technology (CMOS) exhibits low DC current drain. Such systems are thus considered to consume less power and little power when operating. To further reduce power consumption, one known system utilizes a HALT instruction which inhibits processor execution. However, all clock signals utilized by the processor continue to be generated. Since a static microprocessor will maintain its state even in the absence of clock signals, it would be desirable to provide a method for disabling clock signals in an intelligent manner until further processor operations become necessary.

SUMMARY OF THE INVENTION

It is an object of the present invention to provide a method for reducing the power consumed by a static microprocessor.

It is a further object of the present invention to reduce power consumed by static microprocessor by utilizing software instructions which place the microprocessor in a very low current state.

It is yet another object of the present invention to provide a method for reducing power consumed by a CMOS static microprocessor by inhibiting the clock pulses generated until processor operation is required as indicated by some external stimulus.

According to a broad aspect of the invention there is provided a method for reducing, in response to at least one software instruction, energy consumed by a digital system of the type which includes a master oscillator having at least one signal output for producing a clock signal, said method comprising the steps of: decoding said at least one software instruction; and inhibiting said clock signal in response to said software instruction.

The above and other objects, features and advantages of the present invention will be more clearly understood from the following detailed description taken in conjunction with the accompany drawings; in which:

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of a microprocessor in which the present invention may be embodied;

FIG. 2 is a flow diagram illustrating the sequence of operation produced by a WAIT instruction in accordance with the present invention;

FIG. 3 is a flow diagram illustrating the sequence of operation produced by a STOP instruction in accordance with the present invention; and

FIG. 4 is a logic diagram illustrating an apparatus for inhibiting clock signal in response to the WAIT or STOP instructions of FIGS. 2 and 3 to reduce power consumption in the processor.

DESCRIPTION OF THE PREFERRED EMBODIMENT

Referring to FIG. 1, there is shown a block diagram of a microprocessor of a type which may embody the present invention. The microprocessor includes a RAM, timer, and input/output (I/0). The microprocessor has an ALU 11 having an input A, an input B, and a summation output. The summation output is coupled to an 8 bit data bus 12. Data bus 12 couples information to input A. Information is carried to input B by an 8 bit transfer bus 18. An accumulator 14 is coupled to both transfer bus 18 and data bus 12. Accumulator 14 is an 8

bit general purpose register used for arithmetic calculations and data manipulations. An 8 bit index register 16 is coupled between transfer bus 18 and data bus 12. Index register 16 is used during an index mode of addressing and provides an 8 bit address which may be added as an offset to create a new effective address. Index register 16 is also used for calculations and data manipulation during read/modify/write instructions, and as a temporary storage register when not in use for addressing purposes. A transmission gate 17 is used to couple data bus 12 to transfer bus 18. A condition code register 13 is coupled to internal data bus 50 and receives an input from ALU 11. Condition code register 13 is a 5 bit register and contains flags which reflects the results of ALU 11 operations. A first bit contained in condition code register 13 is a carry bit and is set when a carry or a borrow out of ALU 11 occurs during an arithmetic operation. The carry bit can also be modified by certain branch instructions. A second bit in condition code register 13 is a zero bit and is set whenever the result of the last arithmetic, logical, or data manipulation is zero. A third bit is a negative bit which indicates that the result of the last arithmetic, logical, or data manipulation is negative. A fourth bit is a mask interrupt bit and when set, disables both external and timer interrupts. Clearing the interrupt mask bit enables both of the interrupts. Both the timer and external interrupts are latched so that no interrupts are lost because of the interrupt mask bit being set. A fifth bit is a half carry bit, and is set if a carry occurs between bits 3 and 4 of the ALU during an add or an add with carry instruction.

An 8 bit address bus 19 is coupled to data bus 12 by transmission gate 26. Address bus 19 carries the lower 8 bits of an address. The microprocessor of FIG. 1 is capable of addressing up to 8K bytes of external memory with a multiplexed address/data bus. Address bus 19 is coupled to an output buffer 29 and to an incrementer 21. Buffer 29 provides buffered outputs on line 31 which is the 8 bit lower order output address bus. Incrementer 21 is an 8 bit incrementer which can decrement as well as increment. Incrementer 21 is coupled to three registers by line 28. The three registers are an address store register 22, a program counter register 23, and a stack pointer 24. Address store 22 is an 8 bit register which is used to store a lower order effective address such as generated from a branch instruction. Program counter 23 is an 8 bit register which contains the lower 8 bits of a thirteen bit word which is used to point to the next instruction to be executed by the microprocessor. Stack pointer 24 is a 6 bit stack pointer which contains the address of the next free location on a push down/pop up stack. The stack pointer 24 decrements during pushs and increments during pulls. Stack pointer 24 is used to store the location of the return address on subroutine calls and to store the location of the machine state during interrupts. In a preferred embodiment stack pointer 24 is an 8 bit register with the two most significant bits permanently set to a predetermined state.

The outputs of registers 22, 23, and 24 are connected to address bus 19. Program counter 23 also provides an output to data bus 12 via line 27. When one of register 22, 23, or 24 is desired to be modified its contents are transferred by address bus 19 to incrementer 21, where incrementer 21 can increment or decrement the contents, and the contents are then carried by line 28 back to any desired register. This arrangement of incrementer 21 with registers 22, 23 and 24 permit one common incrementer/decrementer for three registers with one of the registers, the program counter 23, also being directly coupled to data bus 12. As mentioned hereinbefore, address bus 19 is coupled to data bus 12 by transmission gate 26.

It should be noted that although the buses are illustrated by one line that they are multiple lines with each different line carrying a different data bit.

The higher five bits of the address word is provided on line 42 by output buffer 41. Output buffer 41 is coupled to a 5 bit address bus 37. It should be noted that in a preferred embodiment, the lower eight bits of the address are multiplexed to external devices while the upper five bits are directly provided on interface pins. Address bus 37 is also coupled to an incrementer/decrementer 38 which is similar to incrementer 21 but only handles five bits. An address store register 32 is coupled from the output of incrementer 38 to address bus 37. Address store 32 contains the high bits of the address while address store 22 contains the lower eight bits of the address. A 5 bit program counter 33 is coupled between the output of incrementer 38 and address bus 37. A 5 bit temporary register 34 is coupled from the output of incrementer 38 to address bus 37. Program counter 33 also provides an output 36 to temporary register 34, which allows the contents of program counter 33 to be directly transferred into temporary register 34. This transfer between registers of course results in faster operation. The output of incrementer 38 is carried by a 5 bit bus or line 39 to registers 32, 33, and 34. Temporary register 34 is directly coupled to data bus 12 by interconnect bus 43. A bit code generator 44 is also connected to data bus 12 which allows any one of the bit lines of data bus 12 to be set or reset under instruction control.

An 8 bit internal data bus 50 is coupled by transmission gate 57 to data bus 12, by a transmission gate 58 to transfer bus 18, and receives information from data bus 12 by buffer/driver 55 and transmission gate 59. Buffer/driver 55 and transmission gate 59 are connected in series. As will be seen hereinafter, the registers are compact, fully static, and are not required to provide static current drive since the drive is provided by buffer 55. By having buffers/drivers 29, 41, and 55 the registers do not require large current drivers and therefore the entire microprocessor can be made smaller in size. An 8 bit I/0 port 47 and an 8 bit I/0 port 48 are coupled to internal data bus 50. I/0 ports 47 and 48 contain data direction registers which control whether the individual interface pins associated with the I/0 ports are serving as an input or an output for the microprocessor. Also coupled to internal data bus 50 is a random access memory (RAM) 49 which stores 8 bit words. In a preferred embodiment, RAM 49 stores 112 bytes. RAM 49 could be used for, among other things, a stack to store the contents of the registers during an interrupt.

Timer 51 is coupled to internal data bus 50 and has a single 8 bit counter with a 7 bit prescaler as its timer. The 8 bit counter is preset under program control and then decrements towards zero. When a zero crossing is detected the timer interrupt request bit of timer 51 is set, then, if a timer interrupt mask and the interrupt mask bit of condition code register 13 are both cleared the microprocessor receives an interrupt. The microprocessor now stores the appropriate registers on the stack, which is located in RAM 49, and then fetches the interrupt address vectors and begins servicing the interrupt. The prescaler of timer 51 is a 7 bit counter used to extend the maximum length of the timer. Timer 51 also provides an

output to input/output logic 53. Input/output logic 53 provides an output to instruction decode and control logic 54. Input/output logic 53 receives and processes a reset and an interrupt request input. An 8 bit instruction register 52 is coupled from internal data bus 50 to instruction decode and control logic 54. Control logic 54 provides decoded instruction outputs and the necessary controls on output lines 56. The outputs on line 56 are used throughout the microprocessor to control the functioning and operation of the microprocessor, a few of such being transmission gates 17, 57, 58, and 59 and bit code generator 44. It is possible for the microprocessor to be a microcomputer simply by the addition of a read only memory (ROM) coupled to internal data bus 50.

A detailed description of the microprocessor shown in FIG. 1 can be found in U.S. patent application Ser. No. 065,294 filed of even date herewith entitled "CMOS Microprocessor Architecture" and assigned to the assignee of the present invention.

As stated previously, a static CMOS microprocessor will maintain its state even in the absence of clock signals. Therefore, to reduce the amount of power consumed, it is desirable to inhibit clock pulses when the processor need not be functioning. This, according to the present invention, is accomplished in two ways. First, a WAIT instruction is added to the instruction repertoire. When executed, the master clock oscillator continues to function as does the timer 51 shown in FIG. 1. However, all other internal processor clocks are inhibited. Thus, the WAIT instruction places the processor in a low power state. The processor may be again rendered operational by (1) activating an external reset, or (2) the presence of an interrupt signal.

A second approach to inhibiting the clocks when the processor need not function is to provide a STOP instruction to the instruction repertoire. When a STOP instruction is executed, both the master clock oscillator and the internal clocks are inhibited. The processor is now in a very low current state; i.e., only leakage current is present. The processor is then restarted as a result of an external reset or interrupt signal; however, it is necessary to provide some period of delay to allow the oscillator to become stable.

FIG. 2 is a flow diagram which illustrates the execution of a WAIT instruction. After a WAIT instruction has been decoded, a latch is set which inhibits the processor clocks. Thus, the processor is placed in a low current mode and awaits either a reset signal or an interrupt signal. If a reset signal is received, the processor will execute a reset routine. If, on the other hand, an interrupt is received, the above referred to latch is reset to enable the processor clocks to commence and an interrupt routine to be executed.

FIG. 3 is a flow diagram illustrating the use of the STOP instruction. After the STOP instruction has been decoded, a second latch is set which turns off the master oscillator. This places the processor in a very low current mode until either a reset or an interrupt signal is received. If an interrupt or a reset signal is received, the second latch is reset to enable the master oscillator. In order to assure that the oscillator is functioning with sufficient logic swing and has settled with respect to frequency, a predetermined amount of delay is provided before the processor executes an interrupt or reset routine.

FIG. 4 is a logic diagram illustrating the apparatus for inhibiting the clocks or master clock oscillator in response to a WAIT or STOP instruction respectively. The instruction is decoded in decode and control logic 54 (FIG. 1), and the individual decoded instruction bits or the complements thereof are applied to gates 92, 94 and 96. The output of NOR gate 92 will be high when signals I0, I2, $\overline{I4}$, $\overline{I5}$, $\overline{I6}$ and I7 are at a logical "1" level. The output of NOR gate 94 will be high when signals $\overline{I0}$, $\overline{I2}$, $\overline{I4}$, $\overline{I5}$, $\overline{I6}$ and I7 are at a logical "1" level. A logical "1" at the output of NOR gate 92 will occur when a WAIT instruction has been decoded, and a logical "1" will appear at the output of NOR gate 94 when a STOP instruction has been decoded. D- type flip-flops 98 and 100 are employed to latch the WAIT and STOP commands respectively. Both flip-flops 98 and 100 are clocked by the output of NOR gate 106 which is applied to the C inputs of flip-flops 98 and 100 directly and which is applied to the $\overline{C}$ inputs of flip-flops 98 and 100 via inverters 102 and 104 respectively. The clocking of these flip-flops occurs at the trailing edge of a load instruction register (LIR) signal which is applied to inverter 108 the output of which is applied to a first input of NOR gate 106. As will be discussed below, the second input of NOR gate 106 is normally low thus permitting flip-flops 98 and 100 to be clocked at the trailing edge of the LIR signal.

The operation of the circuit shown in FIG. 4 will first be described with reference to a WAIT instruction; i.e. a logical "1" appearing at the D input of flip-flop 98. At the trailing edge of the load instruction register signal, flip-flop 98 will be clocked thus latching the WAIT instruction and producing a logical "1" at the Q output of flip-flop 98. This output is coupled to a first input of NOR gate 110 and, as a result thereof, a logical "0" appears at the output of NOR gate 110 and the D input of flip-flop 112.

The master clock oscillator 114 produces outputs 116, 118, 120 and 122 which after propagation through NAND gates 124, 126, 128 and 130 respectively and through inverters 132, 134, 136 and 138 respectively form first and second clock signals (C1 and C2), an address strobe (AS) and a data strobe (DS). Output 116 from master clock 114 is applied directly to the $\overline{C}$ input of flip-flop 112 and to the C input of flip-flop 112 via inverter 140. Therefore, after a logical "0" has been placed at the D input of flip-flop 112 in response to the latching of a WAIT instruction, the Q output of flip-flop 112 will become a logical "0" at the leading edge of the next clock pulse appearing at output 116. The Q output of flip-flop 112 is applied to inputs of NAND gates 124 and 126. Thus, when flip-flop 112 is in the zero state, NAND gates 124 and 126 do not permit passage of the signals appearing on master clock oscillator outputs 116 and 118. As a result, the outputs of inverters 132 and 134 remain a logical "0", and the processor clock pulses C1 and C2 are disabled.

The output of WAIT flip-flop 98 is also applied directly to the D input of flip-flop 142 which is clocked by the output of inverter 113. Thus, when a WAIT instruction is latched, flip-flop 142 is set at the next leading edge of the signal appearing on output 116. The $\overline{Q}$ output of flip-flop 142 is applied to inputs of NAND gates 128 and 130. Since the flip-flop 142 is now set, the $\overline{Q}$ output is a logical "0" thus inhibiting the passage of signals appearing on outputs 120 and 122 of the master clock oscillator. This produces a logical "0" at the output of inverters 136 and 138 thus disabling both the address strobe (AS) and the data strobe (DS) signals.

This is done to prevent the AS and DS lines from charging and discharging unnecessarily during the wait state.

The clock signals and address and data strobe signals will remain inhibited until either a reset signal (RST) or an interrupt signal (INT) is received. The reset signal RST is applied to a first input of NOR gate 146. The output of flip-flop 148 after inversion in inverter 150 is applied to the second input of NOR gate 146. Since flip-flop 148 is normally on, the output of inverter 150 will be a logical "0". Thus, when the reset signal (RST) goes high, a logical "0" will appear at the output of NOR gate 146. This is applied to a first input of NOR gate 152. Clock pulse C2 is applied to a second input of NOR gate 152, and since this clock signal has been disabled as a result of the WAIT instruction, a logical "0" is applied to the second input of NOR gate 152. This results in the production of a logical "1" at the output of NOR gate 152 and at the reset input of WAIT flip-flop 98. With flip-flop 98 now reset, a logical "0" is applied to the D input of flip-flop 142 and a logical "1" is applied to the D input of flip-flop 112. The next clock signal appearing on output 116 of master clock oscillator 114 will cause flip-flop 112 to set and flip-flop 142 to reset. This will enable NAND gates 124, 126, 128 and 130 resulting in the renewed production of clock signals C1 and C2, and address and data strobe signals AS and DS.

If, instead, an interrupt signal (INT), is applied to a first input of NAND gate 154. A mask interrupt signal (MASKI) is applied to inverter 156 the output of which is coupled to a second input of NAND gate 154. Thus, in the presence of an interrupt signal and in the absence of a mask interrupt signal, a logical "0" is applied to the D input of flip-flop 148. Flip-flop 148 is clocked by the output of NOR gate 158 which is applied directly to the C input of flip-flop 148 and to the $\overline{C}$ input of flip-flop 148 via inverter 160. The load instruction register signal (LIR) is applied to a first input of NOR gate 158 and the output of inverter 113 is applied to the second input. Thus, with a zero appearing at the D input of flip-flop 148, the flip-flop is clocked by the next trailing edge of clock 116 resulting in a logical "0" at its Q output. This produces a logical "1" at the output of inverter 150 and a logical "0" at the output of NOR gate 146 since no reset signal is present. With logical "0's" at both inputs of NOR gate 152, a logical "1" is applied to the reset input of WAIT flip-flop 98. Clock signals C1 and C2 and address and data strobe signals AS and DS are again enabled as was described above in the case of a reset signal. It should be apparent that the primary purpose of flip-flop 148 is to synchronize the interrupt signal with the signal appearing on output 116 of the master clock oscillator 114.

When a STOP instruction is decoded, a logical one is placed at the D input of STOP flip-flop 100. This condition is latched by flip-flop 100 at the trailing edge of the load instruction register (LIR) signal as was described previously. The Q output of the STOP flip-flop 100 is applied to a first input of NAND gate 162 and a first input of NOR gate 164. Clock signal C2 is inverted in inverter 166 and applied to a second input of NAND gate 162. This synchronizes the output of NAND gate 162 with processor clock C2. When both of the Q output of STOP flip-flop 100 and the inverted processor clock C2 (i.e. $\overline{C2}$) are high, a logical "1" appears at the output of inverter 168. This output is applied to a reset input of master clock oscillator 114 and to the reset input of flip-flop 112. By resetting the master clock

oscillator in this manner, outputs 116, 118, 120 and 122 are totally disabled.

The logical "1" at the output of STOP flip-flop 100 is also applied to one input of cross-coupled NOR gates 164 and 170. A logical "0" will be produced at the output of NOR gate 164, and a logical "1" will be produced at the output of inverter 172. This produces a logical "0" at the D input of flip-flop 112.

If a reset signal (RST) is received, a logical "1" will appear at the output of NOR gate 152 as was described previously. This output is coupled to the reset input of STOP flip-flop 100. When flip-flop 100 becomes reset, a logical "0" will be applied to the reset input of master clock oscillator 114 thus enabling outputs 116, 118, 120 and 122. However, flip-flop 112 has not yet been set and therefore clock signals C1 and C2 remain disabled. Since output 116 has been enabled, flip-flop 142 is clocked to a reset state thus enabling AS and DS.

Flip-flop 112 becomes set as follows. When the STOP flip-flop 100 was set, a logical "1" appeared at the output of inverter 172 which was applied via line 174 to timer 176. This signal applied to timer 176 enables the timer to count clock pulses received from output 116 of master clock oscillator 114 over line 178. Thus, when the STOP flip-flop 100 is reset, timer 176 begins counting pulses received over line 178. When the counter in the timer reaches a predetermined state, an end of count (EOC) signal is applied to the input of NOR gate 170. The EOC signal is a logical "1" which causes a logical "0" to appear at the output of gate 170 which is in turn applied to an input of NOR gate 164. The output of STOP flip-flop 100 is likewise applied to an input of NOR gate 164. Since the inputs to NOR gate 164 are both zero, the output of NOR gate 164 will be a logical "1" thus producing a logical "0" at the output of inverter 172. With the zeros at both inputs of NOR gate 110, a logical "1" is applied to the D input of flip-flop 112. When the next pulse occurs on output 116 of master clock oscillator 114, flip-flop 112 is clocked producing a logical "1" at its Q output. This now enables gates 124 and 126 to pass the signals appearing on outputs 116 and 118 to produce clock pulses C1 and C2.

Timer 176 may be employed to provide a delay of for example 2 milliseconds. Such timers are well known and a further discussion at this time is not deemed necessary. For example, the timer of U.S. Pat. No. 4,222,103 entitled "Real Time Capture Registers For Data Processor" and assigned to the assignee of the present invention would be suitable.

If instead of a reset signal, an interrupt signal should occur, it is still necessary to reset flip-flop 148. However, since the master clock oscillator 114 has been disabled, no clock pulses can be applied to the C and $\overline{C}$ inputs of flip-flop 148. Therefore, flip-flop 148 must be reset asynchronously. This is accomplished as follows, a logical "1" on the interrupt input is applied to a first input of NAND gate 180. Since clock signal C2 is at a logical "0", a logical "1" is applied to a second input of NAND gate 180 via inverter 182. Finally, a third input of NAND gate 182 is coupled to the output of STOP flip-flop 100 which, after execution of the STOP instruction, is at a logical "1" level. Therefore, the output of NAND gate 180 is at a logical "0" level. This output is inverted by inverter 184 and applied to the R input of flip-flop 148. After flip-flop 148 is reset, the process is the same as above described resulting in the resetting of STOP flip-flop 100.

One additional group of logic comprises NAND gates **186**, **188** and **190**. The first input of NAND gate **186** is coupled to the load instruction register (LIR) signal, and a second input is coupled to the output of inverter **150**. A first input of NAND gate **188** is coupled to the reset signal (RST) and a second input is coupled to the output of NOR gate **110** which is a logical "0" during a WAIT or STOP condition. The outputs of NAND gates **186** and **188** are respectively coupled to first and second inputs of NAND gate **190**. A third input of NAND gate **190** is coupled to the output of NOR gate **110**. The purpose of the output **192** of NAND gate **190** is to load the instruction register with a hardware interrupt thus making the system ready to receive an interrupt.

What is claimed is:

1. In a digital computing system which executes software instructions in synchronization with clock signals generated by a master clock oscillator in an enabled condition thereof, a method for reducing the energy consumed by the digital system, comprising the steps of:
   - decoding a predetermined software instruction selected for execution by said digital computing system;
   - inhibiting passage of said clock signals from said master clock oscillator to said digital computing system in response to the decoding of said predetermined software instruction, and continuing to inhibit passage of said clock signals for a predetermined length of time after said master clock oscillator has been enabled;
   - disabling the generation of said clock signals by said master clock oscillator in response to the decoding of said predetermined software instruction; and
   - enabling the generation of said clock signals by said master clock oscillator in response to a control signal.

2. A method according to claim 1 wherein said control signal comprises an externally produced reset signal.

3. A method according to claim 1 wherein said control signal comprises an interrupt signal.

4. In a digital computing system which executes software instructions in synchronization with clock signals generated by a master clock oscillator in an enabled condition thereof, a method for reducing the energy consumed by the digital system, comprising the steps of:
   - executing a predetermined software instruction selected for execution by said digital computing system;
   - inhibiting passage of said clock signals from said master clock oscillator to said digital computing system in response to the execution of said predetermined software instruction, and continuing to inhibit passage of said clock signals for a predetermined length of time after said master clock oscillator has been enabled;
   - disabling the generation of said clock signals by said master clock oscillator in response to the decoding of said predetermined software instruction; and
   - enabling the generation of said clock signals by said master clock oscillator in response to a control signal.

5. A method according to claim 1 wherein said control signal comprises an externally produced reset signal.

6. A method according to claim 1 wherein said control signal comprises an interrupt signal.

* * * * *

US005084814A

United States Patent [19]

Vaglica et al.

[11] Patent Number: 5,084,814

[45] Date of Patent: Jan. 28, 1992

[54] DATA PROCESSOR WITH DEVELOPMENT SUPPORT FEATURES

[75] Inventors: John J. Vaglica; Jay A. Hartvigsen, both of Austin, Tex.; Rand L. Gray, Hillsboro, Oreg.

[73] Assignee: Motorola, Inc., Schaumburg, Ill.

[21] Appl. No.: 115,479

[22] Filed: Oct. 30, 1987

[51] Int. Cl.[5] G06F 15/78; G06F 9/00

[52] U.S. Cl. 395/325; 364/232.9; 364/232.8; 364/232.3; 364/267; 364/267.91; 364/DIG. 1; 371/16.1; 371/19

[58] Field of Search ... 364/200 MS File, 900 MS File; 371/16-20, 15

[56] References Cited

U.S. PATENT DOCUMENTS

4,153,933	5/1979	Blune, Jr. et al.	364/200
4,231,087	10/1980	Hunsberger et al.	364/200
4,296,469	10/1981	Gunter et al.	364/200
4,307,445	12/1981	Tredennick et al.	364/200
4,312,066	1/1982	Bantz et al.	371/16
4,338,660	7/1982	Kelley et al.	364/200
4,342,078	7/1982	Tredennick et al.	364/200
4,399,870	9/1982	Shaw et al.	364/200
4,441,154	4/1984	McDonough et al.	364/200
4,486,827	12/1984	Shima et al.	364/200
4,524,415	6/1985	Mills, Jr. et al.	364/200
4,667,285	5/1987	Suzuki et al.	364/200
4,670,838	6/1987	Kawata	364/200
4,674,089	6/1987	Poret et al.	371/25
4,677,586	6/1987	Magar et al.	364/900
4,698,750	10/1987	Wilkie et al.	364/200
4,700,293	10/1987	Grone	364/200
4,710,927	12/1987	Miller	371/15
4,740,882	4/1988	Miller	364/200 X

FOREIGN PATENT DOCUMENTS

0129559	8/1983	Japan	371/19

OTHER PUBLICATIONS

Motorola Inc., "16/32-BIT Microcomputer System Components", pp. 1-16 to 1-34, 1985.

Article entitled, "Embedded Bliss for Emulation", by Huffman et al., appearing in Electronic Products, pp. 20-21 on Feb. 1, 1987.

Primary Examiner—Gareth D. Shaw

Assistant Examiner—Paul Kulik

Attorney, Agent, or Firm—Jonathan P. Meyer

[57] ABSTRACT

A data processor with development support features includes an alternate mode of operation in which instructions are received by means of an externally-controlled path. The connections used by the externally-controlled path are not shared by any system resources accessible to the data processor in the normal mode of operation, but are used by other development support features in the normal mode. In a preferred embodiment, an integrated circuit microcomputer includes such a data processor as its CPU. The CPU has access to on-chip peripherals and memory, in addition to off-chip peripherals and memory, in both the normal and alternate modes of operation, by means of a parallel bus which it operates as a bus master. In the alternate mode, the CPU receives instructions by means of a serial bus on which the CPU is a slave device.

30 Claims, 3 Drawing Sheets

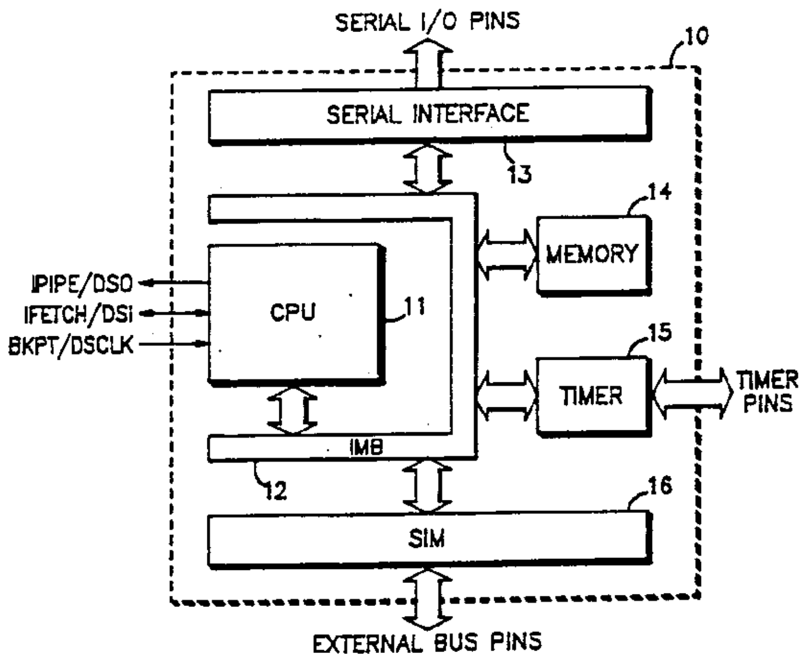


FIG. 1

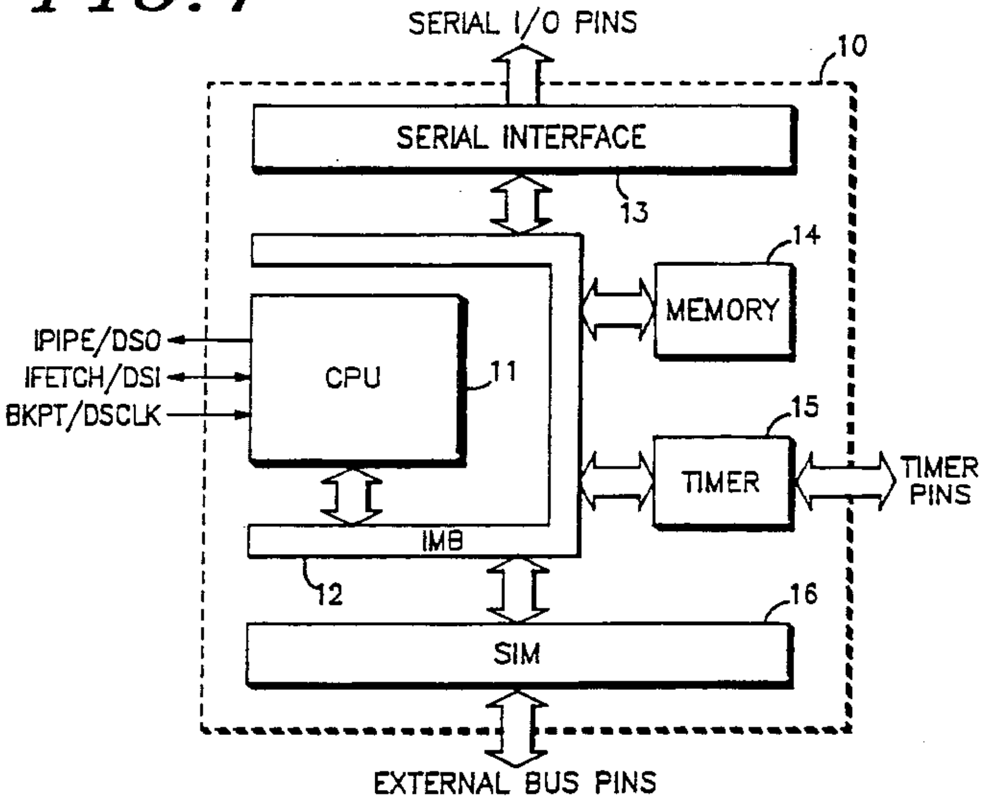

SERIAL I/O PINS
10
SERIAL INTERFACE
13
14
MEMORY
IPIPE/DSO
IFETCH/DSI
BKPT/DSCLK
CPU
11
15
TIMER
TIMER PINS
IMB
12
16
SIM
EXTERNAL BUS PINS

FIG. 4

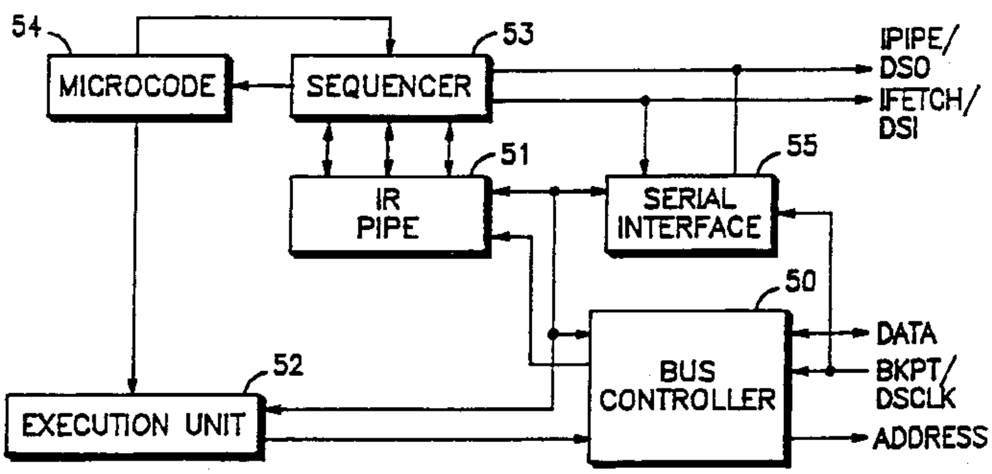
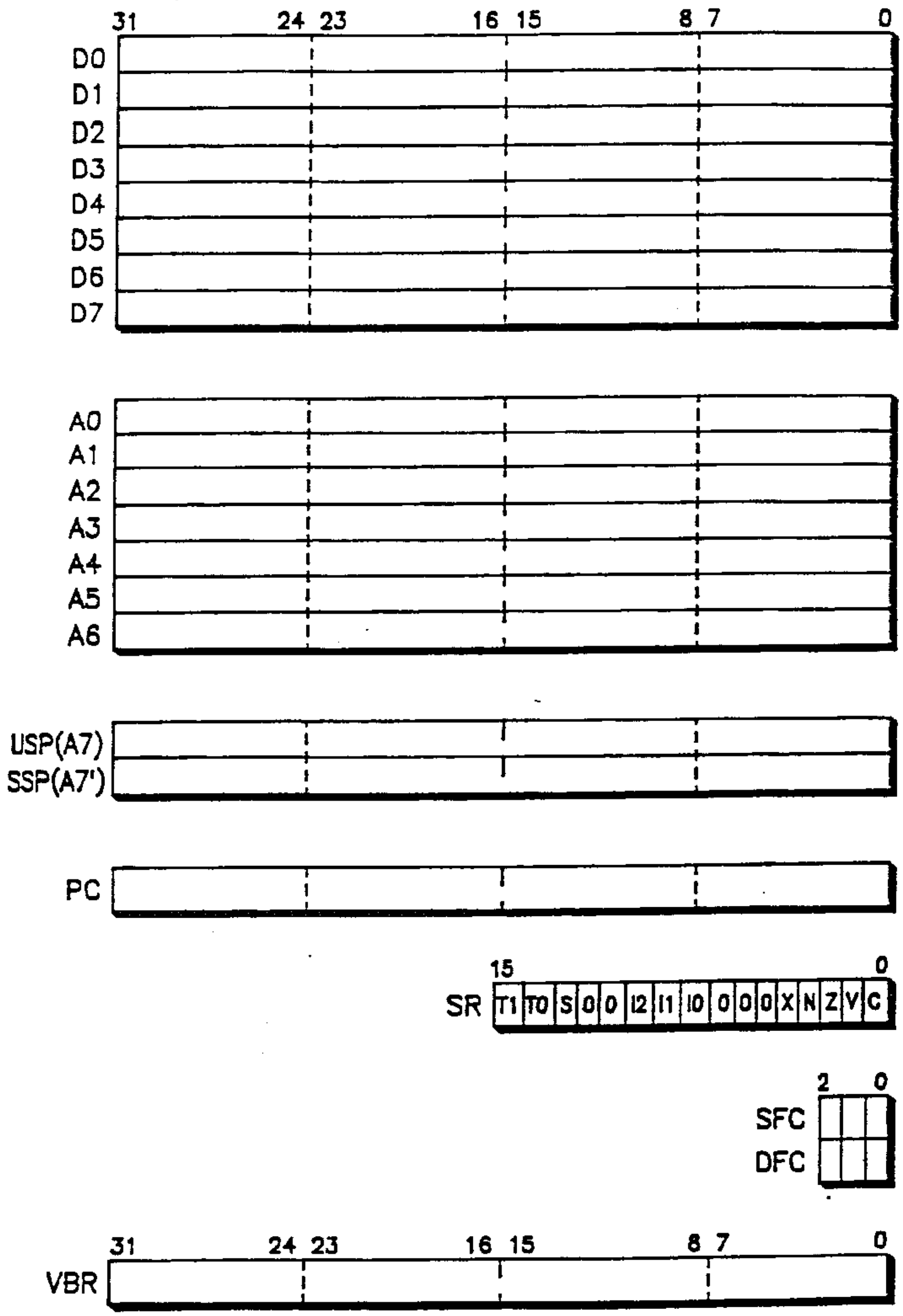

54
MICROCODE
SEQUENCER
53
IPIPE/DSO
IFETCH/DSI
51
IR PIPE
55
SERIAL INTERFACE
50
BUS CONTROLLER
DATA
BKPT/DSCLK
ADDRESS
52
EXECUTION UNIT

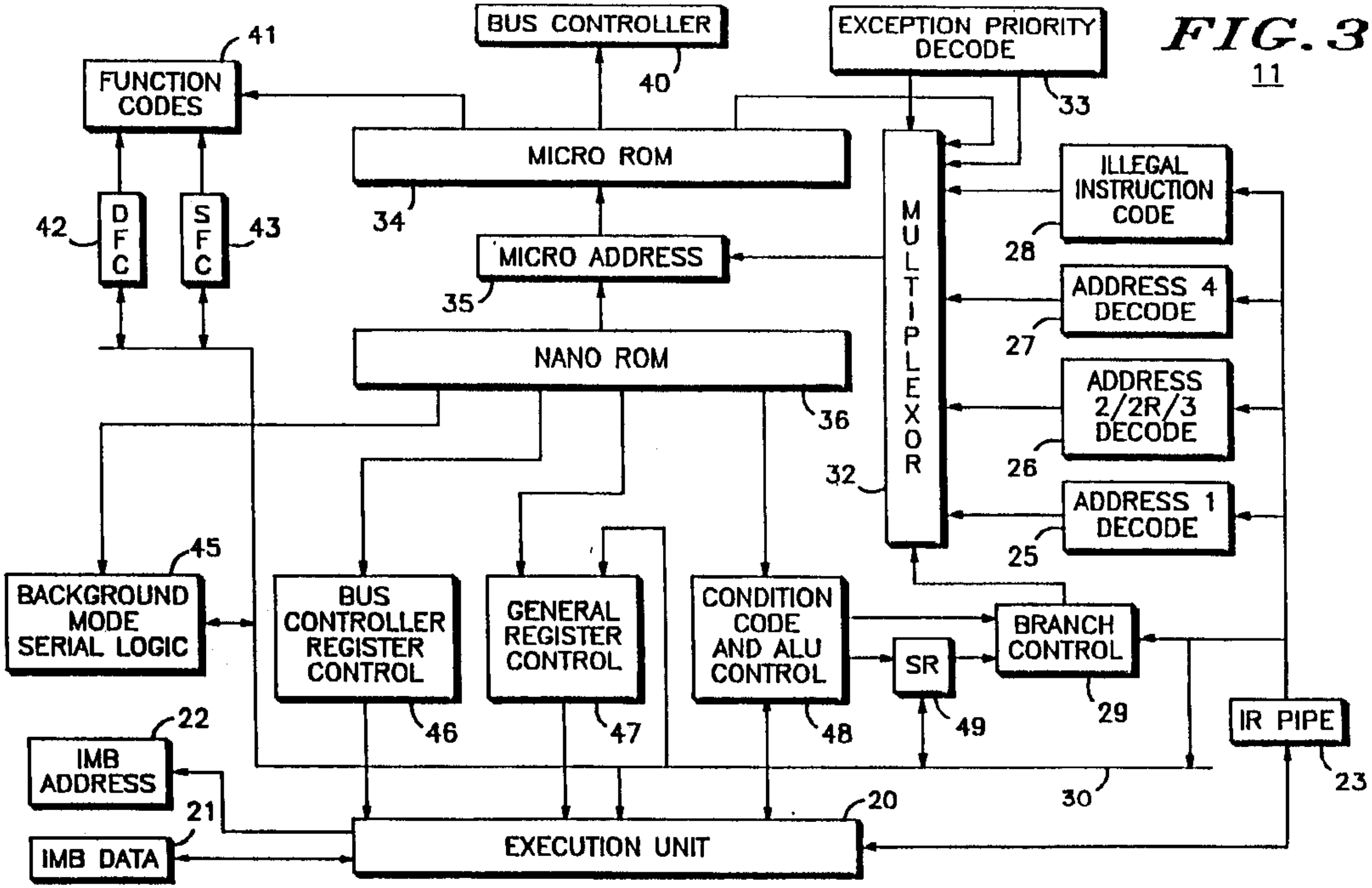
FIG. 2



31
24 23
16 15
8 7
0
D0
D1
D2
D3
D4
D5
D6
D7
A0
A1
A2
A3
A4
A5
A6
USP(A7)
SSP(A7')
PC
15
0
SR
T1
T0
S
0
0
I2
I1
I0
0
0
0
X
N
Z
V
C
2
0
SFC
DFC
31
24 23
16 15
8 7
0
VBR

FIG. 3

11


BUS CONTROLLER
40
EXCEPTION PRIORITY DECODE
33
FUNCTION CODES
41
MICRO ROM
34
DFC
42
SFC
43
MICRO ADDRESS
35
NANO ROM
36
MULTIPLEXOR
32
ILLEGAL INSTRUCTION CODE
28
ADDRESS 4 DECODE
27
ADDRESS 2/2R/3 DECODE
26
ADDRESS 1 DECODE
25
BACKGROUND MODE SERIAL LOGIC
45
BUS CONTROLLER REGISTER CONTROL
46
GENERAL REGISTER CONTROL
47
CONDITION CODE AND ALU CONTROL
48
SR
49
BRANCH CONTROL
29
IR PIPE
23
30
IMB ADDRESS
22
IMB DATA
21
EXECUTION UNIT
20

DATA PROCESSOR WITH DEVELOPMENT SUPPORT FEATURES

FIELD OF THE INVENTION

The present invention relates, in general, to a data processor having features which enhance the functionality and reduce the complexity of development systems intended for use with the system. More particularly, the invention relates to a data processor with a debug mode of operation in which instructions are received by means of a serial interface.

BACKGROUND OF THE INVENTION

Development systems are used in conjunction with data processing systems to assist in the "debugging" of both hardware and software. Typical functions of development systems include the insertion of and response to breakpoints, halting the execution of the data processor to examine and perhaps alter the contents of various system registers and the like and tracing the execution of software.

Data processing systems which are constructed as single integrated circuits, or even as a combination of a few such integrated circuits, present increasingly difficult problems in the design of development systems. As the complexity of the data processor increases, more and more functionality is incorporated onto a single integrated circuit and less access to internal buses, registers and the like is available to development systems.

Prior art integrated circuit data processing systems are known in which a special operating mode is available for debugging and/or emulation purposes. In the special operating mode, the system executes normal instructions as in the normal mode, but fetches the instructions to be executed from a location external to the integrated circuit. In all such systems, one or more of the "system resources" which are utilized by the system in its normal mode of operation (e.g. a communication port) are appropriated for communication with the development system during the special operating mode.

Other development support features which appear in prior art integrated circuit data processing systems include the ability to halt the processor and read certain internal registers and the ability to operate a separate, specially-provided state machine on the same integrated circuit to perform certain debugging functions.

All of the prior art data processing systems with development support features have one or more disadvantages. For instance, the appropriation of system resources for use in the debug mode prevents those resources from being used in their normal fashion, thus requiring extra peripheral circuitry to emulate those functions. The provision of an extra, on-chip debugging machine requires significant extra area on the integrated circuit and is, therefore, suitable only for use on limited numbers of special purpose, debugging-type data processors.

SUMMARY OF THE INVENTION

Accordingly, it is an object of the present invention to provide an improved data processor with development support features.

It is a further object of the present invention to provide a data processor having an alternate mode of operation in which the normal instruction execution apparatus of the processor is used to execute debugging instructions.

It is yet a further object of the present invention to provide a data processor having an alternate mode of operation in which debugging instructions are received for execution by means of a serial communication interface which is not one of the system resources available to the system in the normal mode of operation.

These and other objects and advantages of the present invention are provided by a data processor having a plurality of system resources and comprising: first means for executing instructions; second means for utilizing the system resources in accordance with the execution by said first means of a first plurality of instructions; and third means for providing access to at least one of the system resources in accordance with the execution by said first means of a second plurality of instructions.

These and other objects and advantages of the present invention will be apparent to those skilled in the art from the detailed description below taken together with the drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a block diagram of an integrated circuit data processing system which comprises a preferred embodiment of the present invention;

FIG. 2 is a diagram illustrating the programmer's model of the data processor of FIG. 1;

FIG. 3 is a diagram illustrating the structure of portions of the central processing unit of the data processing system of FIG. 1; and

FIG. 4 is a block diagram illustrating a debug mode serial interface portion of the apparatus of FIG. 3.

DETAILED DESCRIPTION OF THE INVENTION

The terms "assert", "assertion", "negate" and "negation" will be used to avoid confusion when dealing with a mixture of "active high" and "active low" signals. "Assert" and "assertion" are used to indicate that a signal is rendered active, or logically true. "Negate" and "negation" are used to indicate that a signal is rendered inactive, or logically false.

FIG. 1 illustrates an integrated circuit data processing system according to a preferred embodiment of the present invention. A microcomputer **10** comprises a central processing unit (CPU) **11**, an inter-module bus (IMB) **12**, a serial communication interface **13**, on-board memory **14**, a timer module **15** and a system integration module (SIM) **16**. Inter-module bus **12**, which comprises multiple data, address and control signal lines as described in detail below, is coupled to and provides for communication between each of the other components of microcomputer **10**. Serial interface **13** provides for synchronous and/or asynchronous serial data transfer between microcomputer **10** and external devices and systems by means of several serial I/O pins. Memory **14** provides storage space for program and other data useful to microcomputer **10**. Timer module **15** provides various timing functions such as input capture, output compare and the like by means of several timer pins. SIM **16** provides an interface between IMB **12** and an external bus, the details of which are discussed below, and also provides certain system functions such as clock signal generation and distribution.

In addition to its connection to IMB **12**, CPU **11** is coupled to three pins of microcomputer **10** which are

referred to by the labels IPIPE/DSO, IFETCH/DSI and BKPT/DSCLK, respectively. IPIPE/DSO is an output-only pin which provides a signal useful to an external development system for instruction pipeline synchronization (IPIPE) or debug serial communication (DSO), depending on the operating mode of CPU 11. IFETCH/DSI is a bidirectional pin which is useful to an external development system for instruction pipeline synchronization (IFETCH) or debug serial communication (DSI), depending on the operating mode of CPU 11. BKPT/DSCLK is an input-only pin which is useful to an external development system for directing CPU 11 to execute a breakpoint (BKPT), to enter a debug mode of operation or for debug serial communication (DSCLK), depending on the operating mode of CPU 11. Each of these functions is described in more detail below.

While not all of the development support features of microcomputer 10 are the subject of the present invention, all are inter-related in the context of the operation of the preferred embodiment. The development support features of microcomputer 10 include: a "background debug" mode of operation, trackability of the instruction pipeline of CPU 11, external visibility of IMB bus cycles and the ability of both on-board modules and external devices to insert breakpoints.

The background debug mode of operation involves toggling CPU 11 into a mode in which execution of normal instructions fetched via IMB 12 is halted and execution proceeds with special debugging instructions which are received by means of a serial communication interface comprising the IPIPE/DSO, IFETCH/DSI and BKPT/DSCLK pins. The special debugging instructions are capable of reading and writing the CPU's registers (including several registers not accessible in the normal mode of operation), reading and writing locations in the memory map (which includes locations in memory 14 and registers in serial interface 13, timer module 15 and SIM 16) and resuming normal instruction execution.

The ability to track the internal instruction pipeline of CPU 11 involves the use of the IPIPE/DSO and IFETCH/DSI pins. In normal mode, CPU 11 asserts IPIPE for one clock cycle when the first stage of the pipe is advanced and for two clock cycles when both stages of the pipe are advanced. Similarly, IFETCH is asserted for a single clock cycle to indicate that the current bus cycle is an instruction fetch and for two clock cycles to indicate that the pipe has been flushed.

The ability to make purely internal bus cycles on IMB 12 visible outside of microcomputer 10 involves the setting of control bits in SIM 16 to a particular value. When these bits are properly set, IMB cycles are echoed on the external bus. These external bus cycles are distinguished from normal external bus cycles by the fact that the address strobe (AS) is never asserted.

The breakpoint features of microcomputer 10 allow either an external device or one of the on-board peripherals to insert a breakpoint associated with an access of any particular memory location. Access to an operand at that location or execution of an instruction stored at that location will cause CPU 10 to execute a breakpoint exception handling routine (or to enter background debug mode).

All of these features will be explained in greater detail below.

In order to provide an adequate background for the discussion to follow, the signals of both IMB 12 and the external bus must be described. The tables below provide a summary of those signals and their definitions. Most of these signals and their definitions are similar to the bus signals of microprocessor available commercially from Motorola, Inc. of Austin, Tex. An example of such a microprocessor is the MC68010.

	INTER-MODULE BUS SIGNALS		
SIGNAL NAME	MNEMONIC	FUNCTION	DIRECTION
Address Bus	ADDR0-ADDR31	32 bit address bus capable of addressing 4 Gbytes	output
Data Bus	DATA0-DATA15	16 bit data bus capable of 8 and 16 bit transfers	input/output
Function Code	FC0-FC2	Identifies CPU state (supervisor/user) and address space of current bus cycle	output
Clock	CLOCK	Master system clock	input
Cycle Start	CYS	Indicates start of internal bus cycle	input/output
Address Strobe	AS	Indicates second phase of bus cycle and that address is valid	output/input
Data Strobe	DS	Indicates third phase of bus cycle that data is valid for write cycle	output
Read/Write	WRITE	Defines a bus cycle as a read or write, relative to the bus master	output
Trabsfer Size	SIZ0-SIZ1	Specifies the number of bytes yet to be transferred in a bus cycle	output/input
Data Transfer Acknowledge	DTACK	Slave response which terminates a bus cycle	input
Bus Error	BERR	Provides for termination of a bus cycle if no valid response is received	input
Relinquish and Retry	RRT	Provides a means for breaking a bus mastership standoff at the internal/external bus boundary	input
Retry	RETRY	Provides for termination of a bus cycle which should be rerun	input
Halt	HALT	Indicates that the CPU has halted due to an abnormal condition	output
Breakpoint Request	BKPT	Signals a request for a breakpoint on the current bus cycle	input
Breakpoint Acknowledge	FREEZE	Indicates that the CPU has entered background debug mode	output
System Reset	SYSRST	Provides a "soft" reset which does not disturb system configuration data	output
Master Reset	MSTRST	Provides a "hard" reset of everything	input
Interrupt Request Level	IRQ1-IRQ7	Prioritized interrupt requests to the CPU	input
Autovector	AVEC	Specifies that autovector feature is to be used during an interrupt ack. cycle	input
Bus Request	BR0-BRn	Prioritized bus	input

-continued

INTER-MODULE BUS SIGNALS			
SIGNAL NAME	MNEMONIC	FUNCTION	DIRECTION
		mastership arbitration	signals
Bus Lock	BLOCK	Allows bus master to retain bus	input/output
Test Mode	TSTMOD	Enables test mode for all devices	input
Enable IMB test lines	IMBTEST	Changes function of IRQ1-IRQ7 to test lines	input

Note that signal directions in the table above are specified with respect to CPU 11.

EXTERNAL BUS SIGNALS			
SIGNAL NAME	MNEMONIC	FUNCTION	DIRECTION
Address Bus	A0-A23*	24 bit address bus capable of addressing 16 Mbytes	input/output
Data Bus	D0-D15	16 bit data bus capable of 8 and 16 bit transfers	input/output
Function Code	FC0-FC2*	Identifies CPU state (supervisor/user) and address space of current bus cycle	input/output
Boot Chip Select	CSBOOT	Programmable chip select for boot-up	output
Bus Request	BR*	Bus Mastership Request line	input/output
Bus Grant	BG*	Bus Mastership Grant line	output
Bus Grant Acknowledge	BGACK*	Bus Mastership Grant Acknowledge line	input/output
Data and Size Acknowledge	DSACK0-DSACK1	Indicates that data is valid on read and that data has been received on write. Also indicates port size.	input/output
Address Strobe	AS	Indicates that address, function codes etc., are valid	output/input
Data Strobe	DS	Indicates that data is valid on write and that slave should drive data on read	input/output
Read/Write	WRITE	Defines a bus cycle as a read or write, relative to the bus master	input/output
Transfer Size	SIZ0-SIZ1	Indicates single or multi-byte transfer	input/output
Bus Error	BERR	Provides for termination of a bus cycle if no valid response is received	input/output
Halt	HALT	Indicates that the CPU has halted due to an abnormal condition	input/output
Interrupt Request Level	IRQ1-IRQ7	Prioritized interrupt requests to the CPU	input
Autovector	AVEC	Specifies that autovector feature is to be used during an interrupt ack. cycle	input
Bus Lock	BLCK	Indicates indivisible bus cycles	input/output
Reset	RESET	System Reset	input/output
External RAM Chip Enable	RAMCE	Useful to disable external devices during resets	output
External System Clock	CLK	External system clock - bus clock	output
Crystal Osc.	EXTAL, XTAL	Pins for connection of external osc. or clock circuit	input/output
External Filter Capacitor	XFC	Allows connection of external filter cap. to internal clock	output
Synthesizer VDD	VDDSYN	Provides power to internal clock synthesizer	input
Freeze	FRZ/QUOT	Acknowledges entry into background mode and outputs quotient bit in test mode	output
Test Mode Enable and Tri-State Control	TSTME/TSC	Triggers test mode and causes output drivers to be tri-stated	input
Clock Mode Select	MODCK	Selects source of system clock	input

The pins denoted above with an asterisk, the address pins A19-A23, the function code pins FC0-FC2, the bus request pin BR, the bus grant pin BG and the bus grant acknowledge pin BGACK may also be used as programmable chip select pins. This feature of microcomputer 10 is not related to an understanding of the present invention. Signal directions are specified with respect to microcomputer 10.

FIG. 2 illustrates the programmer's model of CPU 11, that is, those registers which are accessible to software executing on CPU 11. CPU 11 has eight, 32-bit wide data registers, labeled D0-D7. Similarly, CPU 11 has access to seven, 32-bit wide address registers, labeled A0-A6. Each of these registers is both readable and writable while executing normal instructions.

CPU 11 has two stack pointers: a user-stack pointer labeled USP and a supervisor stack pointer labeled SSP. The stack pointers are used to point to the top of a "stack" of memory locations in which are stored information about the state of CPU 11 which is useful for returning from exception processing and the like. In addition to the labels USP and SSP, the current stack pointer may be accessed by referencing an eighth address register, labeled A7 (or A7'). Which stack pointer is accessed in this wasy is determined by the current user/supervisor state of CPU 11. The terms user and supervisor refers to the fact that CPU 11 has two "privilege states" of those names which control whether certain "privileged" instructions are executable.

The program counter of CPU 11 is a 32-bit wide register labeled PC. As is conventional, the program counter contains the address in memory at which the next instruction to be executed is stored.

The status register of CPU 11, labeled SR, inludes a separately addressable condition code register (CCR) which forms the lower byte of the status register. The lowest five bits of the status registers are the familiar carry, overflow, zero, negative and extend condition code bits, as is the case with all Motorola 68000-family microprocessors. The next three bits of the status register are unimplemented and read as zeros. Bits 8, 9 and 10 are the interrupt mask bits used by CPU 11 to determine the lowest priority interrupt which is currently un-

masked. Bits 11 and 12 are unimplemented and read as zero. Bit 13 of SR is the supervisor bit, which indicates that CPU 11 is in the supervisor privilege state if it is set. The two highest order bits of the status register enable tracing of instruction execution, as in the Motorola 68020 microprocessor.

CPU 11 has two three-bit registers used in certain circumstances to determine the states of the function code outputs (FC0–FC2). These are the SFC and DFC (for source function code and destination function code) registers. These registers are used by the MOVES instruction to move data between the various address spaces of CPU 11, as is the case with the Motorola 68010 and 68020 microprocessors. As is described more fully below, SFC and DFC are also used to provide the function code outputs while in the background debug mode.

The final register appearing in the programmer's model is VBR, which stands for vector base register. VBR is used as a base for calculating vectors which are used to locate the starting addresses of exception handling software routines. This register is also present in the 68010 and 68020 microprocessors.

All of the registers described above are, in one way or another, accessible while CPU 11 is executing normal instructions. Some, such as the stack pointers and the upper half of the status register, can only be accessed while CPU 11 is in the supervisor privilege state. There are other registers, such as the temporary registers denoted by ATEMP and BTEMP, which do not appear in the programmer's model. These registers are affected by normal program execution, but are not accessible in the sense that they cannot be read or written to.

FIG. 3 is a block diagram illustrating the detailed structure of CPU 11 of FIG. 1. Of necessity, FIG. 3 is a somewhat simplified view of a very complex structure. Detailed descriptions of very similar CPU structures are to be found in U.S. Pat. Nos. 4,342,078, 4,296,469, 4,307,445 and 4,524,415. All of these patents are commonly owned with the present invention. The listed patents are hereby incorporated herein by reference.

CPU 11 is a pipelined, microprogrammed data processor. An execution unit 20 is connected bidirectionally to a data bus portion 21 of IMB 12 (see FIG. 1). Execution unit 20 is also coupled to address portion 22 of IMB 12. Execution unit 20 is coupled bidirectionally to an instruction register (IR) pipe 23, which is a multistage pipeline through which each normal instruction passes prior to and while it is actually being executed.

Execution unit 20 contains all of the arithmetic, shift, register and other logic necessary to execute each of the normal and special debug instructions of CPU 11. Except for the destination function code and source function code registers and the status register, all of the registers described above with reference to FIG. 2 may be thought of as residing within execution unit 20.

Outputs of IR pipe 23 are coupled to inputs of an address 1 decoder 25, an address 2/2R/3 decoder 26, an address 4 decoder 27, an illegal instruction decoder 28, branch controller 29 and a general internal (G) bus 30. Address decoders 25, 26 and 27 operate on portions of instructions to produce a next micro address. Branch controller 29 produces a portion of a next micro address. Illegal instruction decoder 28 also produces a next micro address. Each of these next micro addresses, or portions thereof, is input to a multiplexor 32. Multiplexor 32 also receives inputs from an exception priority

decoder 33 and a micro ROM 34. The output of multiplexor 32 is input to a micro address register 35.

Micro address register 35 contains the micro address of the micro instruction currently being executed. Micro address register 35 provides outputs to both micro ROM 34 and to nano ROM 36. In combination, micro ROM 34 and nano ROM 36 produce the control outputs necessary to execute each instruction and to proceed to the next instruction. An output of micro ROM 34 is coupled to bus controller 40, which operates IMB 12 as a bus master to fetch normal instructions and to read and write data.

Another output of micro ROM 34 is coupled to function code logic 41, which also receives inputs from destination function code register 42 and source function code register 43. Function code logic 41 produces the function code bits which indicate the address space of each IMB bus cycle. DFC 42 and SFC 43 are also bidirectionally coupled to G bus 30.

Various outputs of nano ROM 36 are coupled to background mode serial logic 45, bus controller register control logic 46, general register control logic 47 and condition code and ALU (arithmetic and logic unit) control logic 48. General register control logic 47 also receives inputs from G bus 30. Background mode serial logic 45 is bidirectionally coupled to G bus 30. Bus controller register control 46 and general register control 47 provide outputs to execution unit 20. Condition code and ALU control logic 48 is bidirectionally coupled to execution unit 20 and has an output coupled to branch control 29. Status register 49 is bidirectionally coupled to G-bus 30, receives inputs from condition code and ALU control logic 48 and provides outputs to branch control logic 29. G bus 30 provides inputs to execution unit 20.

While executing normal instructions, CPU 11 operates in a fashion substantially similar to the data processors described in the above-referenced U.S. Patents. Instructions are fetched from memory through the operation of bus controller 40, passed through IR pipe 23 and executed by execution unit 20. When toggled into the background debug mode however, normal instruction execution is halted. Debug instructions received by means of background mode serial logic 45 are passed through IR pipe 23 and executed. Since the same instruction execution apparatus is used in both the normal and background modes, very little additional hardware is required to implement the background debug mode. Relatively small additions to one or more of address decoders 25, 26 and 27 and to micro ROM 34 and nano ROM 36 are required, in addition to background mode serial logic 45 itself.

FIG. 4 illustrates in greater detail the logical relationships between the major components of CPU 11 as they relate to the subject matter of the present invention. In the normal mode of operation, a bus controller 50 fetches instructions and data operands from memory by means of a data bus and an address bus (which are merely the data and address portions of IMB 12 of FIG. 1), which are operated by bus controller 50 as a bus master. Instructions are passed from bus controller 50 to IR pipe 51 and data operands are passed to an execution unit 52. Various outputs from the multiple stages of IR pipe 51 are provided to a sequencer 53, which provides inputs to microcode 54. Microcode 54 also provides inputs to sequencer 53. The outputs of microcode 54 provide inputs to control the operation of execution unit 52. As is apparent, this description is a somewhat more

simplified view of the operation of a pipelined, microcoded processor than was presented above.

Sequencer 53 controls the advancement of instructions through the stages of IR pipe 51. It provides the output signals IPIPE and IFETCH so that an external development system can continuously monitor the contents of IR pipe 51. In the preferred embodiment, IR pipe 51 may be considered to be a three stage pipe. Sequencer 53 asserts IPIPE for one clock cycle when the first stage of the pipe is advanced to the second stage. Sequencer 53 asserts IPIPE for two clock cycles when both the first stage is advanced to the second stage and the second stage is advanced to the third stage. IFETCH is asserted for one clock cycle when the current bus cycle is an instruction fetch (i.e., when the results of the bus cycle will be loaded into IR pipe 51). IFETCH is asserted for two clock cycles when IR pipe 51 is cleared, or, as is commonly said, flushed. This occurs, for instance, when a program branch is taken or exception processing is commenced. The assertion of IFETCH for two cycles also indicates that the data returned on the current bus cycle will be the first word loaded into the newly cleared IR pipe 51.

A BKPT input to bus controller 50 includes both the BKPT signal line of IMB 12 and the direct BKPT input to CPU 11 as shown in FIG. 1. Any device having access to BKPT may monitor the bus cycles on IMB 12 and/or the external bus and insert breakpoints whenever the address being accesed falls within some preselected range, or at any other time deemed appropriate. If the bus cycle on which the BKPT input to bus controller 50 is asserted is not an instruction fetch cycle, the breakpoint will be acknowledged immediately upon the completion of the execution of the currently executing instruction. If the cycle is an instruction fetch, then the breakpoint will be made "pending" and only acknowledged after the instruction fetched on that cycle is eventually executed.

Insertion of a breakpoint by means of either the IMB BKPT line or the external BKPT line is referred to as a "hardware" breakpoint. In addition to a hardware breakpoint, a "software" breakpoint is provided in the preferred embodiment. The software breakpoint is substantially the same as the software breakpoint provided on the Motorola 68020 microprocessor and simply comprises one or more of the normal instructions which are responded to as breakpoint instructions.

Upon either a hardware or a software breakpoint, a special bus cycle referred to as a breakpoint acknowledge cycle is executed. This bus cycle is terminated by the receipt of either a BERR signal or a data transfer and size acknowledge signal on DSACKO-1. On both hardware and software breakpoints, termination of the cycle by BERR causes CPU 11 to proceed with exception processing. An illegal instruction exception is taken in the case of a software breakpoint and a breakpoint exception is taken in the case of a hardware breakpoint. If a breakpoint acknowledge cycle initiated by a software breakpoint is terminated by the receipt of a data transfer and size acknowledge on the DSACKO-1 lines, the returned data is used to replace the breakpoint instruction and execution resumes with the replacement instruction. If a breakpoint acknowledge cycle initiated by a hardware breakpoint is terminated by receipt of a data transfer and size acknowledge on DSACKO-1, the returned data is ignored and execution is resumed with the next instruction as if the breakpoint were never received. Hardware breakpoints also comprise one of several methods of entering the background debug mode, if it is enabled.

Three different methods for entry into the background debug mode are provided in the preferred embodiment. However, none of the entry methods is successful unless the background debug mode is enabled. Enablement of the background debug mode is determined when microcomputer 10 is reset by means of the external RESET signal. When microcomputer 10 exits reset, the state of the external BKPT pin is latched. If the latched value of BKPT is a logical zero, then background debug mode entry is enabled. Otherwise, entry is disabled. The state of enablement of background debug mode entry is not alterable unless microcomputer 10 is reset again.

Assuming that background debug mode entry is enabled, the mode may be entered by assertion of the BKPT input to bus controller 50, either by one of the devices connected to IMB 12 or by an external device via the external BKPT pin, by execution of the background instruction which is provided as one of the normal instructions, or by double bus faults (i.e., a second bus fault occuring during the processing of an exception caused by a first bus fault).

The action of the various breakpoint and background debug mode entry methods is summarized in the following table.

EVENT	RESPONSE	
	BGM ENABLED	DISABLED
double bus fault	background mode	halt
hardware breakpoint	background mode	breakpoint ack
software breakpoint	breakpoint ack	breakpoint ack
background instr.	background mode	illegal exc.

The background debug mode of operation of the apparatus of FIG. 4 involves the termination of normal program execution. Once debug mode is entered, interrupts to CPU 11 are not acknowledged or acted upon and instruction tracing is disabled. Debug instructions and any address and/or operand data required for the execution thereof are provided to IR pipe 51 by serial interface 55. Serial interface 55 receives these instructions and data from an external development system via the DSI pin and simultaneously passes status and result data to the development system via the DSO pin. Serial interface 55 operates in a manner similar to the familiar Motorola Serial Peripheral Interface in which one bit of data is shifted in and one bit of data is shifted out on each cycle of the clock signal DSCLK. Since DSCLK is provided by the external development system, it controls the serial interface and is said to be the master thereof. Serial interface 55 operates as a slave.

The choice to operate serial interface 55 as a slave when in the background debug mode is a feature of the preferred embodiment of the present invention. This choice provides a greater degree of freedom in the implementation of a development system designed for use with this invention, since the development system is not constrained by the clock rate at which CPU 11 is operating when in its normal mode.

Upon the occurence of one of the background debug mode entry events, the IMB FREEZE signal is asserted to indicate background debug mode entry to other devices on IMB 12. The external bus FREEZE signal is also asserted. No context stacking is performed for

entry into background debug mode. Normal instruction execution is halted and serial interface 55 is enabled.

Appropriate response to FREEZE by peripheral devices provides for entry into, and recovery from, background debug mode without apparent loss of "real time" context. For instance, a watchdog timer which might form a part of SIM 16, could be configured to generate an interrupt or reset periodically if it is not cleared before it times out. Obviously, if such a timer is turned off by disabling its clock in response to FREEZE, normal instruction execution could begin without losing the state of the watchdog timer as it appeared prior to background debug mode entry. The most appropriate response to FREEZE must be determined on an individual basis for each component of a system.

An external development system which is to interact with CPU 11 during background debug mode must, of course, monitor the FREEZE line of the external bus. When FREEZE is asserted, the external development system will release BKPT (if it asserted BKPT) and commence transferring debug instructions and data.

Once it has been placed in background debug mode, CPU 11 simply waits for a debug instruction to be supplied by an external development system by means of serial interface 55. A single "transfer" of information via serial interface 55 comprises 17 bits: 16 bits of instruction, operand or response and one bit of status information. Note that 17 bits are transferred in each direction; that is, from serial interface 55 to the external development system and from the external development system to serial interface 55.

Obviously, the first transfer after entry into background debug mode comprises a debug instruction to be executed. Subsequent transfers may involve the provision of operands such as addresses and data necessary for debug instruction execution and/or the provision of results of debug instruction execution to the external development system.

Once a complete debug instruction, and any necessary address or data is shifted into serial interface 55, the instruction is executed. In the preferred embodiment, serial interface 55 passes instructions to be executed through a stage of IR pipe 51 which is always empty upon background debug mode entry. This is possible since such mode entry occurs only on instruction boundaries.

As mentioned above, the 17 bits received by the external development system on each transfer are intended to be interpreted as 16 bits of data and one status bit (the most significant bit). If the status bit is 0, then the data is to be interpreted as a valid transfer of the results of the excecution of the previous debug instruction. If the status bit is 1 and the data bits are all 0, the development system is to infer that CPU 11 is not yet ready to respond to the previous debug instruction. If the status bit is 1 and the least significant four data bits are also 1, then the previously transferred debug command was either an illegal command or was not received properly. If the status bits and all of the data bits are 1, then execution of the debug instruction was terminated by a bus error. The development system may re-synchronize the serial interface by transferring 17 or more consecutive 1's. Except when re-synchronizing, the external development system must set the most significant bit of data transferred into serial interface 55 to zero.

Commands in the set of debug instructions include: 1) read an address or data register, 2) write an address or data register, 3) read a system register, 4) write a system register, 5) read a memory location, 6) write a memory location, 7) dump a memory block, 8) fill a memory block, 9) flush the instruction pipe and resume normal execution, 10) assert the SYSRST line of IMB 12 to reset peripherals and 11) perform a no-operation (NOP).

In the case of register instructions, the source or destination register is specified by an encoded bit field of the instruction itself. All address and data registers of the programmer's model are accessible. In addition, all of the system registers of the programmer's model are accessible (including PC, SR, USP, SSP, SFC, DFC and VBR). In addition, the ATEMP and BTEMP registers which do not appear in the programmer's model are accesible via the read system register and write system register debug instructions. Of course, any other registers which do not appear in the programmer's model could also be made accessible to debugging instructions, if desired. On register writes, the 32 bits comprising the two transfers immediately following the instruction supply the data to be written.

In the case of memory reads and writes, the address is specified by 32 bits of address supplied by the two transfers immediately following the instruction. For writes, the byte, word or long word to be written is supplied by the one or two transfers following the address. For memory reads, the SFC register supplies the function codes to be used; that is, the bits of SFC determine the address space to be accessed. Similarly, for memory writes, the DFC register supplies the function codes to be used. This implies that SFC and DFC must be properly set up using the write system register commands before using the memory read and write commands.

The memory block read and write instructions are used when more than a single operand from memory is to be accessed. Either instruction must be preceded by a read or write memory instruction in order to set the starting address for the block operation. Once the starting address has been set, subsequent block reads and writes need only supply the operand size and the operand itself on writes.

Note that the ability to perform memory reads and writes implies that IMB 12, the external bus interface and the other on-chip peripherals must still be active in the background debug mode of operation. That is, they do not respond to the FREEZE signal by halting operation. FREEZE is provided primarily to allow subsystems such as watchdog timers, periodic interrupt generators and the like to respond in an appropriate manner to maintain the context of the machine as may be desired upon resumption of normal instruction execution, as mentioned above.

Other possible debug mode instructions which may be useful include an instruction to restart normal instruction execution without flushing the instruction pipe and an instruction to return information about the particular version of microcomputer 10 being used, such as a microcode revision number. It should be noted that the read and write memory commands can be used to access memory-mapped registers which control the on-chip peripherals of microcomputer 10. For instance, a register within SIM 16 contains a two bit field which controls the external visibility of bus cycles of IMB 12. These bits are readable and writable in the background debug mode.